Physics-informed learning of governing equations from scarce data

Zhao Chen, Yang Liu & Hao Sun

Bruno J. Zorzet

sinc(i)

sinc(i) - Santa Fe

sinc(i) - Santa Fe

Physics-informed learning of governing equations from scarce data

Zhao Chen, Yang Liu & Hao Sun

Bruno Zorzet

CONICET

UNL

sinc(i)

Introduction

Why PINNs?

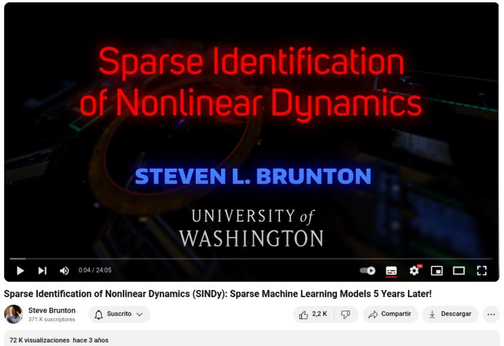
- Incorporating domain knowledge
- Data scarcity
- Solving inverse problems
- Simulation and modeling

Why PINNs?

- Incorporating domain knowledge
- Data scarcity
- Solving inverse problems
  - Inferring Unknown Coefficients
  - Inferring Boundary or Initial Conditions
  - Learning Hidden Forces or Fields
  - **Discovering the Entire PDE**
- Simulation and modeling

Background Researches: SINDy

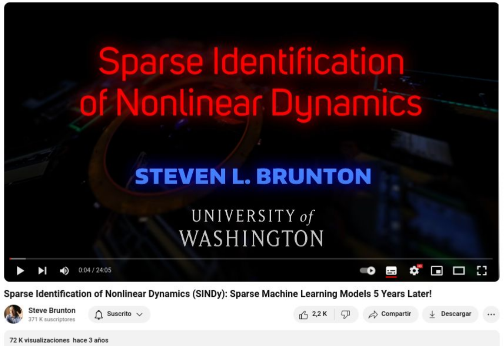
- SINDy (Sparse Identification of Nonlinear Dynamics)



Brunton, S.L., Proctor, J.L., and Kutz, J.N. (2016b). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci., 113(15), 3932–3937.

Background Researches: SINDy

- SINDy (Sparse Identification of Nonlinear Dynamics)

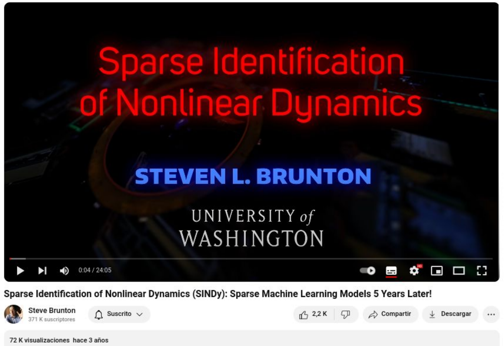


Advantages

Simple and interpretable set of equations (ordinary differential equations).

Brunton, S.L., Proctor, J.L., and Kutz, J.N. (2016b). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci., 113(15), 3932–3937.

Background Researches: SINDy



- SINDy (Sparse Identification of Nonlinear Dynamics)

Advantages

Simple and interpretable set of equations (ordinary differential equations).

Limitations

Struggles with noisy and incomplete data

Difficult with partial differential equations (PDEs).

Brunton, S.L., Proctor, J.L., and Kutz, J.N. (2016b). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci., 113(15), 3932–3937.

Main purpose of the paper

They address these limitations by incorporating physics-informed neural networks (PINNs) and sparse regression into a single framework to improve robustness and handle noisy

Main purpose of the paper

They address these limitations by incorporating physics-informed neural networks (PINNs) and sparse regression into a single framework to improve robustness and handle noisy

Root-Branch Network Architecture

Alternating Direction Training Strategy

Main purpose of the paper

They address these limitations by incorporating physics-informed neural networks (PINNs) and sparse regression into a single framework to improve robustness and handle noisy

Root-Branch Network Architecture

Multiple datasets

Handling with differents boundaries and initial conditions

Alternating Direction Training Strategy

Main purpose of the paper

They address these limitations by incorporating physics-informed neural networks (PINNs) and sparse regression into a single framework to improve robustness and handle noisy

Root-Branch Network Architecture

Alternating Direction Training Strategy

Train the DNN and the sparse matrix coefficients

Summary of Contributions

- A novel **“root-branch” network** that handles multiple datasets with different initial/boundary conditions.

Summary of Contributions

- A novel **"root-branch" network** that handles multiple datasets with different initial/boundary conditions.
- An alternating direction **training strategy** to optimize both neural network parameters and sparse PDE coefficients, improving convergence and efficiency.

Summary of Contributions

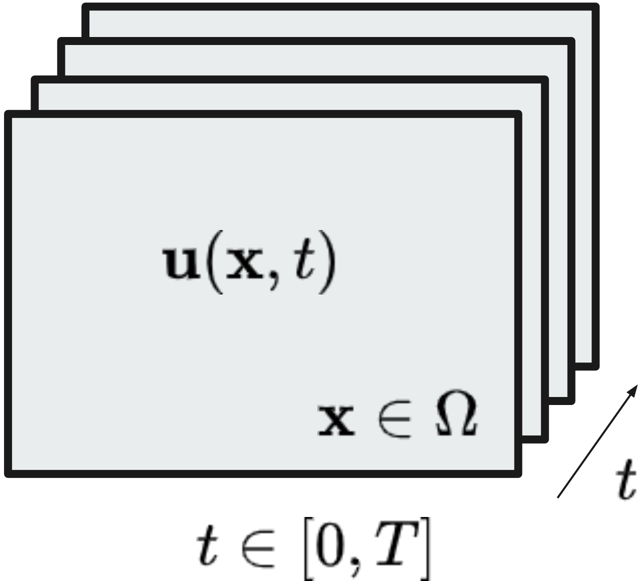
- A novel **"root-branch" network** that handles multiple datasets with different initial/boundary conditions.
- An alternating direction **training strategy** to optimize both neural network parameters and sparse PDE coefficients, improving convergence and efficiency.
- Improved **robustness and generalizability** by combining the strengths of PINNs (accurate derivative calculation) with sparse regression (interpretability), making it suitable for noisy and incomplete data.

Methods

Modeling Systems

$$\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}$$

$\mathbf{u}(\mathbf{x}, t) \longrightarrow$ Physical magnitude

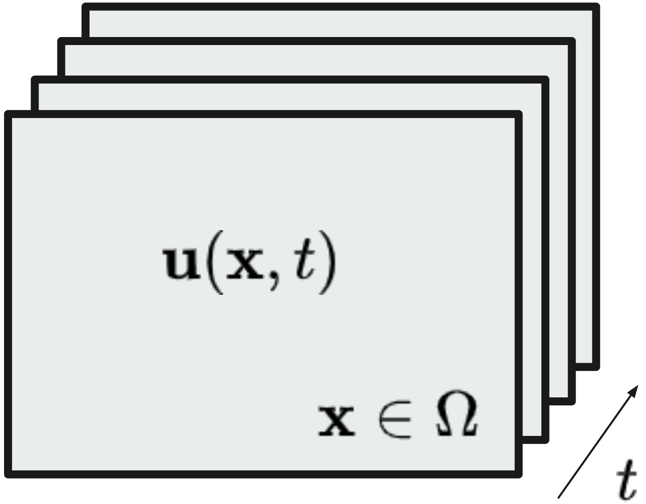


Modeling Systems

$$\boxed{\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}}$$

Assumptions

$$\mathbf{u} = \mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^{1 \times n}$$



$t \in [0, T]$

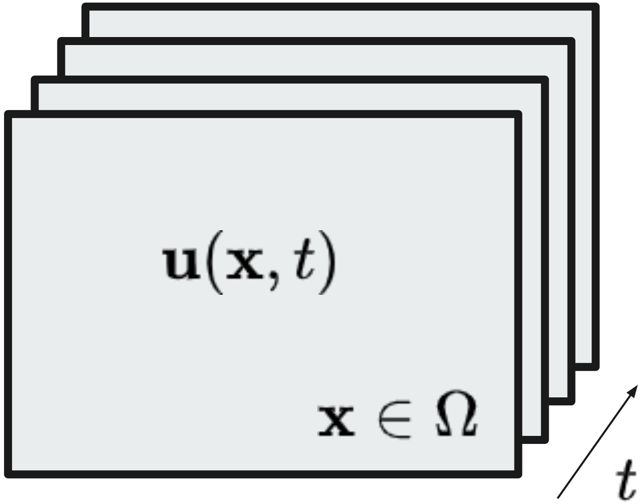
Modeling Systems

$$\boxed{\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}}$$

Assumptions

$$\mathbf{u} = \mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^{1\times n}$$

$$\boxed{\mathbf{p} = \mathbf{p}(\mathbf{x}, t) = \mathbf{0}}$$



$t \in [0, T]$

Modeling Systems

$$\boxed{\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}}$$

Assumptions

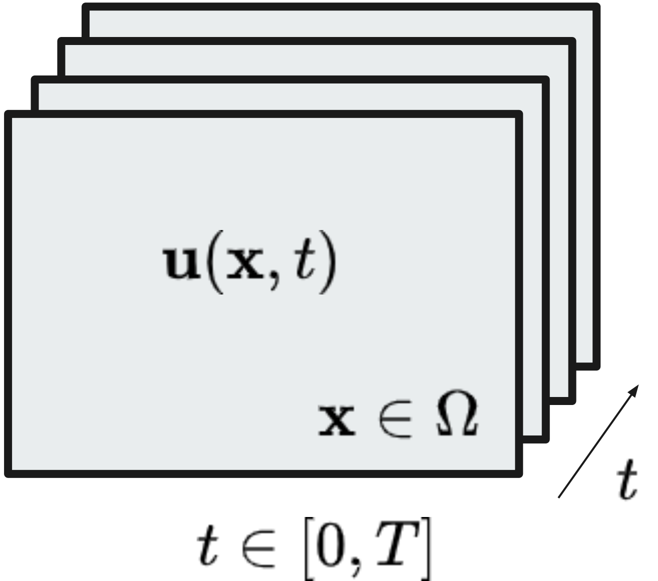
$$\mathbf{u} = \mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^{1\times n}$$

$$\boxed{\mathbf{p} = \mathbf{p}(\mathbf{x}, t) = \mathbf{0}}$$

Boundaries and initials conditions

$$\mathcal{I}[\mathbf{x} \in \Omega,\ t = 0;\ \mathbf{u};\ \mathbf{u}_t] = \mathbf{g}(\mathbf{x})$$

$$\mathcal{B}[\mathbf{x} \in \partial\Omega;\ \mathbf{u};\ \nabla_{\mathbf{x}} \mathbf{u}] = \mathbf{h}(\mathbf{t})$$



Modeling Systems

$$\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}$$

$$\boxed{\mathbf{p} = \mathbf{p}(\mathbf{x}, t) = \mathbf{0}}$$

Modeling Systems

$$\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}$$

$$\boxed{\mathbf{u}_t = \phi\Lambda}$$

Modeling Systems

$$\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}$$

$$\boxed{\mathbf{u}_t - \phi\Lambda = 0}$$

Modeling Systems

$$\boxed{\mathbf{u}_t - \phi\Lambda = 0}$$

Assumptions

$$\mathbf{u} = \{u,\ v,\ w\}$$

$$\phi = \{1, \mathbf{u}, \mathbf{u}^2, \ldots, \mathbf{u}_x, \mathbf{u}_y, \ldots, \mathbf{u}^3 \odot \mathbf{u}_{xy}, \ldots, \sin(\mathbf{u}), \ldots\} \in \mathbb{R}^{1\times s}$$

$$\Lambda = [\lambda^u,\ \lambda^v,\ \lambda^w] \in \mathbb{R}^{s\times 3}$$

Modeling Systems

$$\boxed{\mathbf{u}_t - \phi\Lambda = 0}$$

In some part of the presentations we try to estimate Λ

Modeling Systems

$$\boxed{\mathbf{u}_t - \phi\Lambda = 0}$$

In some part of the presentations we try to estimate Λ

$$\phi\Lambda \begin{cases} \text{Parsimonious} \\ \text{Closed} \end{cases}$$

Modeling Systems

$$\boxed{\mathbf{u}_t - \phi\Lambda = 0}$$

In some part of the presentations we try to estimate Λ

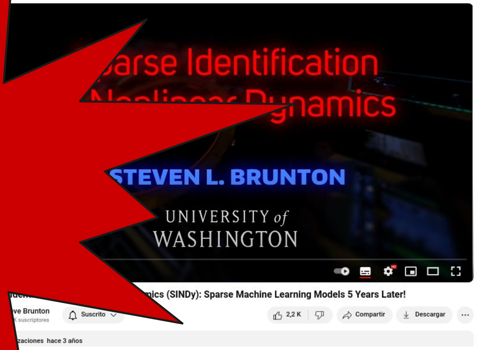
$$\phi\Lambda \left\{ \begin{array}{lll} \text{Parsimonious} & = & \text{Simple} \\ \text{Closed forms} & = & \text{Completely defined} \end{array} \right.$$

Training Datasets

Background Researches: SIND

- SINDy (Sparse Identification of Non

REMEMBER



Advantages

Simple and interpretable set of equations (ordinary differential equations).

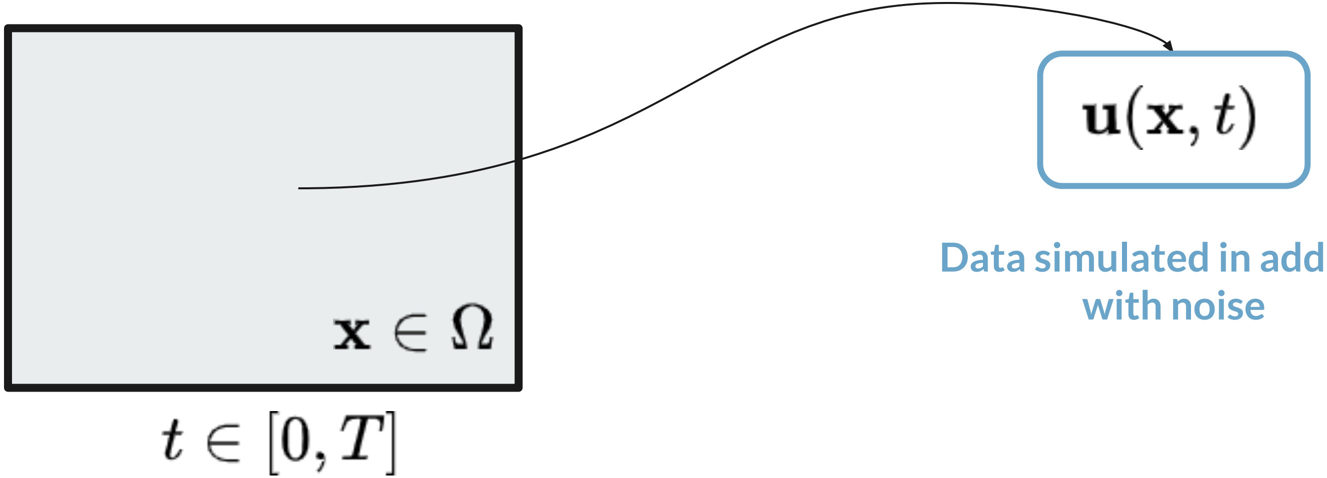
Limitations

Struggles with noisy and incomplete data

Difficult with partial differential equations (PDEs).

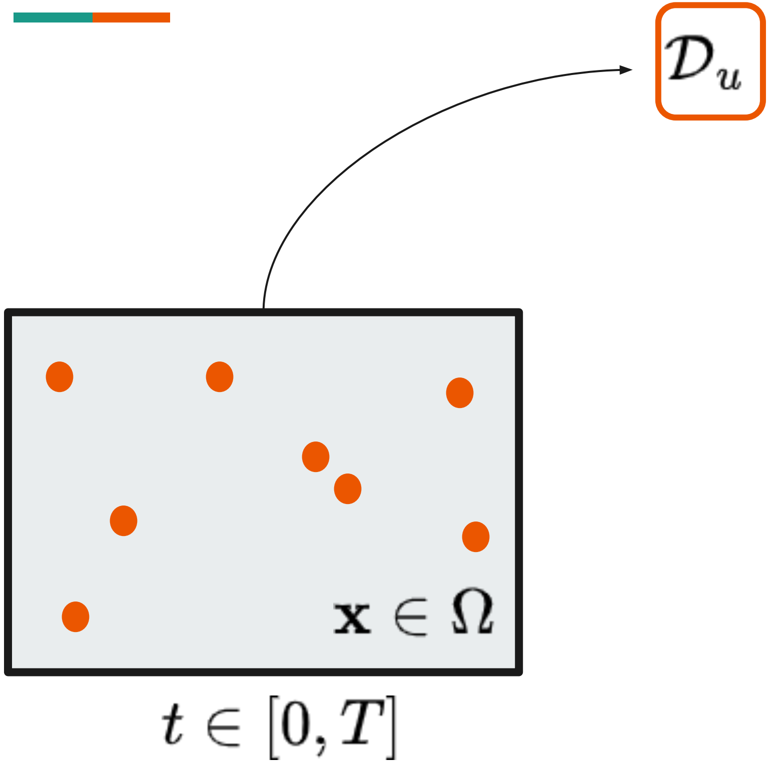
Brunton, S.L., Proctor, J.L., and Kutz, J.N. (2016b). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci., 113(15), 3932–3937.

Training Datasets

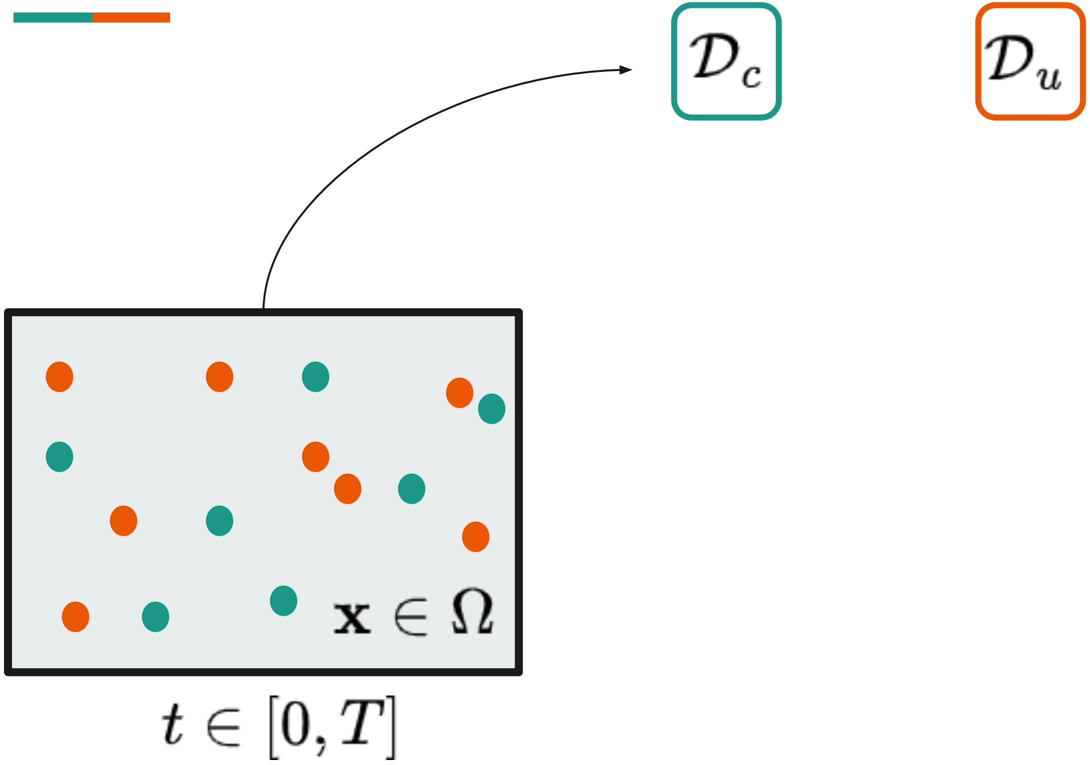


Data simulated in addition
with noise

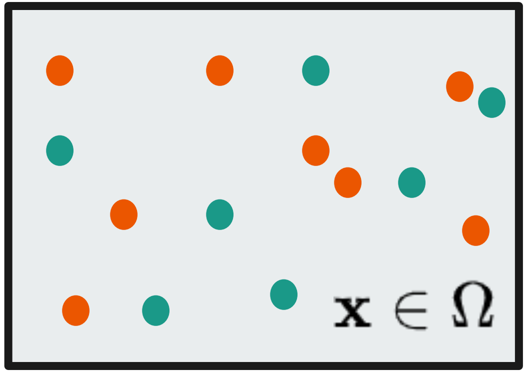
Training Datasets



Training Datasets



Training Datasets



$t \in [0, T]$

$\mathcal{D}_c$

Collocation points for compute the physical loss

$\mathcal{D}_u$

Measurement points for compute the physical loss

What we try to do?

What we try to do?

t

x

y

z

What we try to do?

t

x

y

z

ESTIMATE $\mathbf{u}$

$$\hat{u}(\mathbf{x}, t; \theta)$$
$$\hat{v}(\mathbf{x}, t; \theta)$$
$$\hat{w}(\mathbf{x}, t; \theta)$$

What we try to do?

t

x

y

z

ESTIMATE $\mathbf{u}$

$$\hat{u}(\mathbf{x}, t; \theta)$$
$$\hat{v}(\mathbf{x}, t; \theta)$$
$$\hat{w}(\mathbf{x}, t; \theta)$$

COMPUTE ϕ

$$\mathbf{u}_t - \phi \boxed{\Lambda} = 0$$

Estimation of physical magnitude

$$\begin{pmatrix} t \\ x \\ y \\ z \end{pmatrix} \longrightarrow \boxed{\begin{matrix}\text{DNN} \\ \mathcal{N}(\theta)\end{matrix}} \longrightarrow \begin{pmatrix} \hat{u}(\mathbf{x}, t; \theta) \\ \hat{v}(\mathbf{x}, t; \theta) \\ \hat{w}(\mathbf{x}, t; \theta) \end{pmatrix}$$

Input

Model

Output

$\mathcal{D}_u$

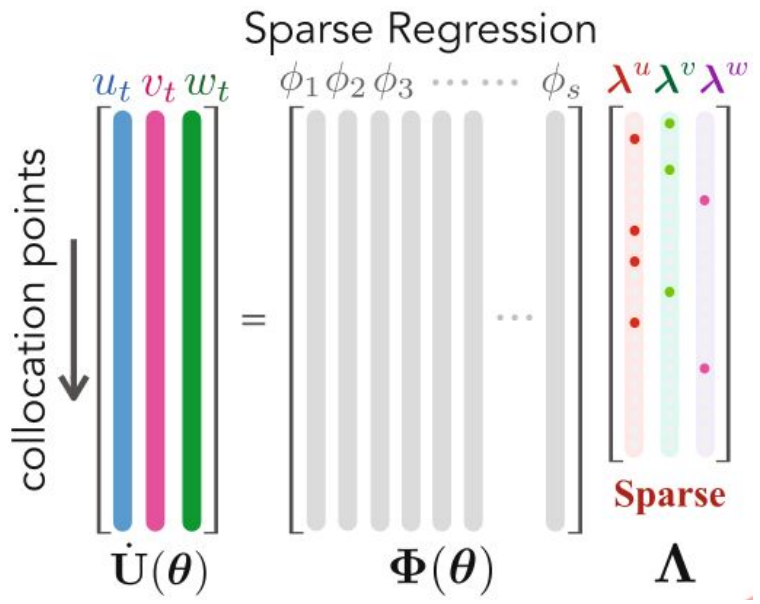
Measurement points

Estimation of physical law

$$\hat{u}(\mathbf{x}, t; \theta)$$
$$\hat{v}(\mathbf{x}, t; \theta)$$
$$\hat{w}(\mathbf{x}, t; \theta)$$

Compute the derivatives terms

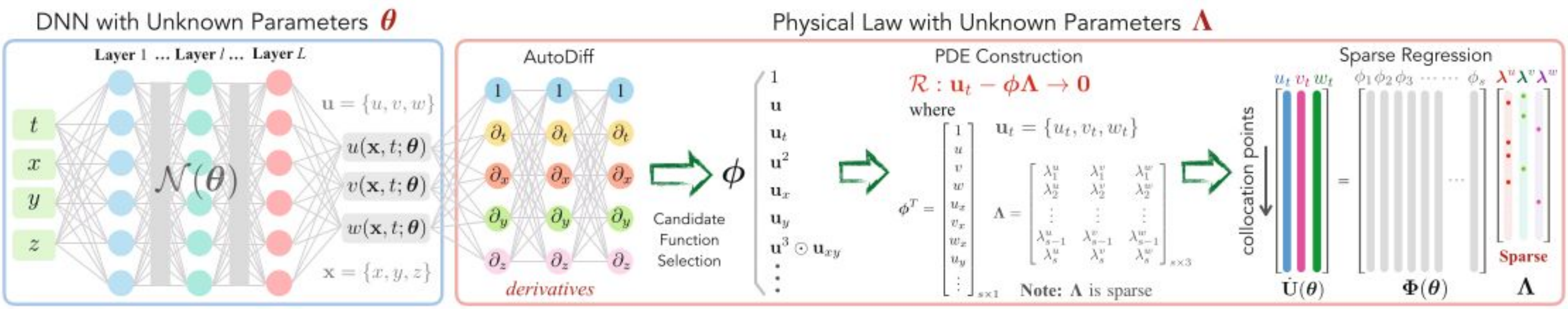
$$\phi^T = \begin{bmatrix} 1 \\ u \\ v \\ w \\ u_x \\ v_x \\ w_x \\ u_y \\ \vdots \end{bmatrix}$$



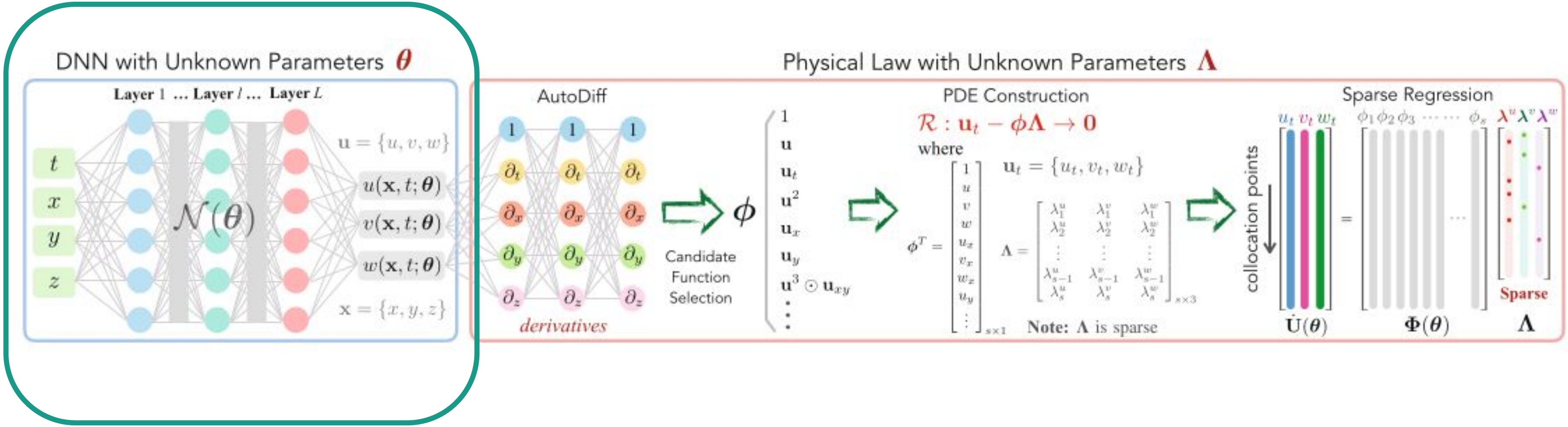
$\mathcal{D}_c$

Collocation points

Entire Pipeline

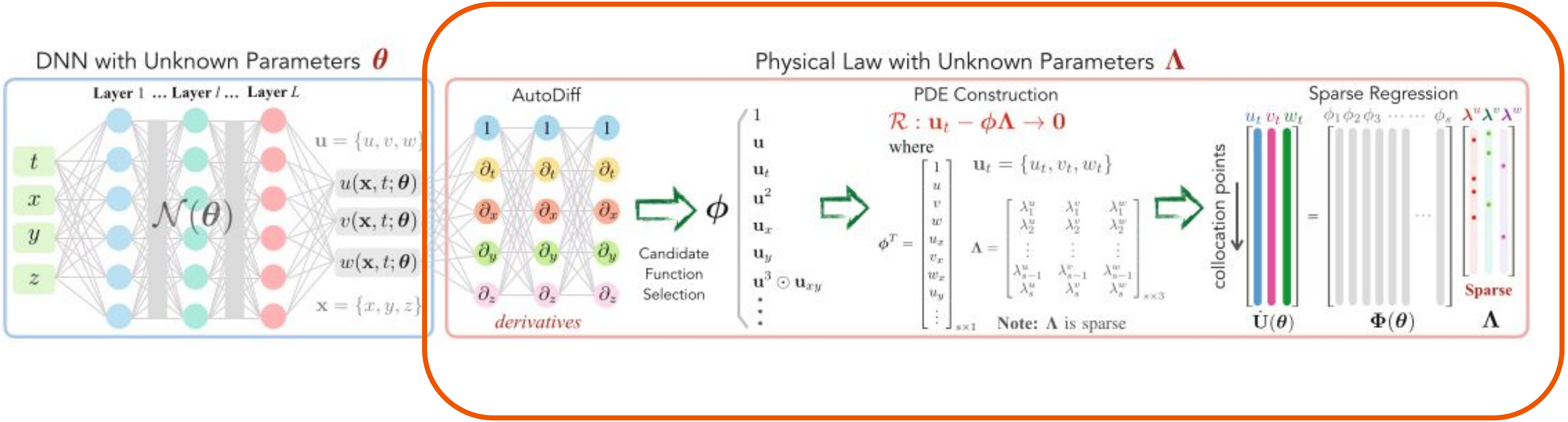


Entire Pipeline



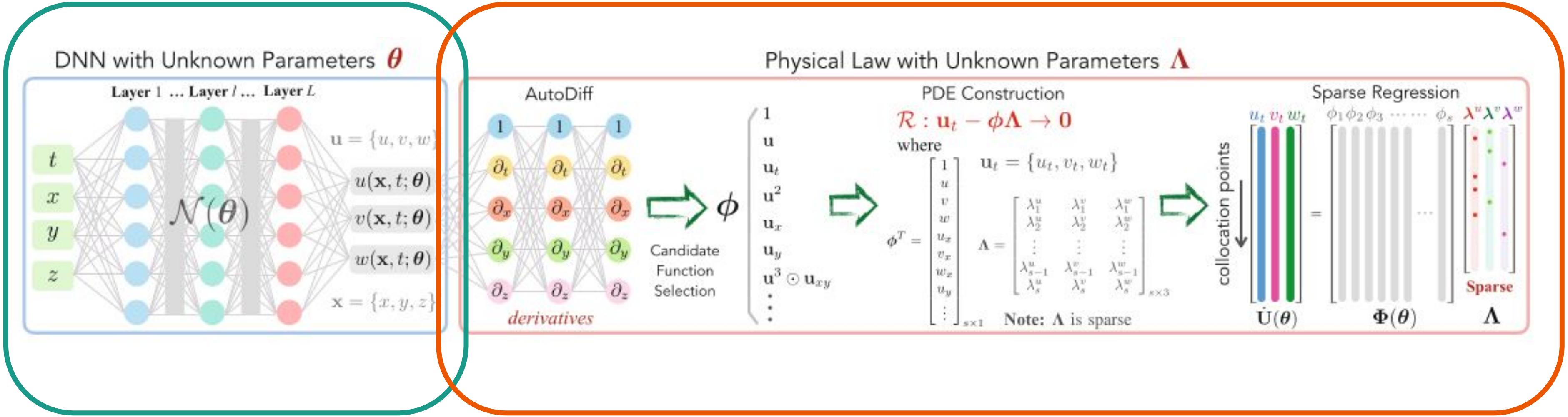
Estimation of the physical magnitude

Entire Pipeline



Estimation of the PDE

Entire Pipeline



Estimation of the physical magnitude

Estimation of the PDE

Loss Function

$$\mathcal{L}(\theta,\ \Lambda;\ \mathcal{D}_u,\ \mathcal{D}_c) = \mathcal{L}_d(\theta;\ \mathcal{D}_u) + \alpha\, \mathcal{L}_p(\theta,\ \Lambda;\ \mathcal{D}_c)\ +\ \beta\|\Lambda\|_0$$

Loss Function

$$\mathcal{L}(\theta,\ \Lambda;\ \mathcal{D}_u,\ \mathcal{D}_c) = \boxed{\mathcal{L}_d(\theta;\ \mathcal{D}_u)} + \alpha\, \mathcal{L}_p(\theta,\ \Lambda;\ \mathcal{D}_c)\ +\ \beta \|\Lambda\|_0$$

Loss Function

$$\mathcal{L}(\theta,\ \Lambda;\ \mathcal{D}_u,\ \mathcal{D}_c) = \boxed{\mathcal{L}_d(\theta;\ \mathcal{D}_u)} + \boxed{\alpha\, \mathcal{L}_p(\theta,\ \Lambda;\ \mathcal{D}_c)} + \ \beta\|\Lambda\|_0$$

Loss Function

$$\mathcal{L}(\theta,\ \Lambda;\ \mathcal{D}_u,\ \mathcal{D}_c) = \boxed{\mathcal{L}_d(\theta;\ \mathcal{D}_u)} + \boxed{\alpha\, \mathcal{L}_p(\theta,\ \Lambda;\ \mathcal{D}_c)} + \boxed{\beta \|\Lambda\|_0}$$

Loss Function

$$\mathcal{L}(\theta,\ \Lambda;\ \mathcal{D}_u,\ \mathcal{D}_c) = \boxed{\mathcal{L}_d(\theta;\ \mathcal{D}_u)} + \boxed{\alpha\, \mathcal{L}_p(\theta,\ \Lambda;\ \mathcal{D}_c)} + \boxed{\beta \|\Lambda\|_0}$$

$$\boxed{\mathcal{L}_d(\theta;\ \mathcal{D}_u) = \frac{1}{N_m} \left\| \mathbf{u}^\theta - \mathbf{u}^m \right\|_2^2}$$

$$\boxed{\mathcal{L}_p(\theta, \Lambda;\ \mathcal{D}_c) = \frac{1}{N_c} \left\| \dot{\mathbf{U}}(\theta) - \Phi(\theta)\Lambda \right\|_2^2}$$

Loss Function

$$\mathcal{L}(\theta,\ \Lambda;\ \mathcal{D}_u,\ \mathcal{D}_c) = \boxed{\mathcal{L}_d(\theta;\ \mathcal{D}_u)} + \boxed{\alpha\,\mathcal{L}_p(\theta,\ \Lambda;\ \mathcal{D}_c)} + \boxed{\beta\|\Lambda\|_0}$$

$$\boxed{\mathcal{L}_d(\theta;\ \mathcal{D}_u) = \frac{1}{N_m}\left\|\mathbf{u}^\theta - \mathbf{u}^m\right\|_2^2}$$

$$\boxed{\mathcal{L}_p(\theta, \Lambda;\ \mathcal{D}_c) = \frac{1}{N_c}\left\|\boxed{\dot{\mathbf{U}}(\theta)} - \Phi(\theta)\Lambda\right\|_2^2}$$

Alternating Optimization Algorithm

1. Pre-trained the network using Adam + L-BFGS

Alternating Optimization Algorithm

1. Pre-trained the network using Adam + L-BFGS

$$\{\theta^\star,\ \Lambda^\star\} = \arg\min_{\{\theta,\Lambda\}}\{\mathcal{L}_d(\theta;\ \mathcal{D}_u) + \alpha\,\mathcal{L}_p(\theta,\ \Lambda;\mathcal{D}_c) + \gamma\|\Lambda\|_1\}$$

Alternating Optimization Algorithm

1. Pre-trained the network using Adam + L-BFGS

$$\{\theta^{\star},\ \Lambda^{\star}\} = \arg\min_{\{\theta,\Lambda\}}\{\mathcal{L}_d(\theta;\ \mathcal{D}_u) + \alpha\,\mathcal{L}_p(\theta,\ \Lambda;\mathcal{D}_c) + \gamma\|\Lambda\|_1\}$$

2. Iteration:
   I. Compute the outputs of the net and derivatives terms
   II. Optimize Λ via STRidge (sequential threshold ridge regression)
   III. Train the network via Adam + L-BFGS.

Alternating Optimization Algorithm

$\mathcal{D}_{train}$

Pretraining

ADO

Loop of training:

1. Compute the outputs of DNN for collocation points
2. Estimate the matrix sparse matrix via STRIDge (LOOP)
3. Train the DNN with this new estimation of matrix sparse using physical loss and data loss

Alternating Optimization Algorithm

Algorithm 1 The proposed ADO for network training: $[\boldsymbol{\theta}_{\text{best}}, \boldsymbol{\Lambda}_{\text{best}}] = \texttt{ADO}(\mathcal{D}_u, \mathcal{D}_c, \alpha, \gamma, \Delta\delta, n_{\max}, n_{\text{str}})$

1: **Input:** Measurement data $\mathcal{D}_u$, collocation points $\mathcal{D}_c = \{\mathbf{x}_i, t_i\}_{i=1,2,\dots,N_c}$, relative weighting of loss functions α and γ, threshold tolerance increment $\Delta\delta$ for STRidge, maximum number of ADO iterations $n_{\max}$, and maximum number of STRidge cycles n_{str}.
we take a 2D system in a 2D domain as an example: $\mathbf{u} = \{u, v\}$ and $\mathbf{x} = \{x, y\}$

2: Split measurement data $\mathcal{D}_u$ into training-validation sets ($n_{\text{tr}}/n_{\text{va}} = 80/20$): $\mathcal{D}_u^{\text{tr}} \in \mathbb{R}^{n_{\text{tr}} \times 2}$ and $\mathcal{D}_u^{\text{va}} \in \mathbb{R}^{n_{\text{va}} \times 2}$. # $N_m = n_{\text{tr}} + n_{\text{va}}$

3: Split collocation points $\mathcal{D}_c$ into training-validation sets ($m_{\text{tr}}/m_{\text{va}} = 80/20$): $\mathcal{D}_c^{\text{tr}} \in \mathbb{R}^{m_{\text{tr}} \times 3}$ and $\mathcal{D}_c^{\text{va}} \in \mathbb{R}^{m_{\text{va}} \times 3}$. # $N_c = m_{\text{tr}} + m_{\text{va}}$

4: Initialize the *Tensor Graph* for the entire network.

5: Pre-train the network via combined Adam and L-BFGS with $\{\mathcal{D}_u^{\text{tr}}, \mathcal{D}_c^{\text{tr}}\}$, and validate the trained model with $\{\mathcal{D}_u^{\text{va}}, \mathcal{D}_c^{\text{va}}\}$, namely,

$$\{\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Lambda}}_0\} = \arg\min_{\{\boldsymbol{\theta}, \boldsymbol{\Lambda}\}} \left\{\mathcal{L}_d(\boldsymbol{\theta}; \mathcal{D}_u) + \alpha\mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\Lambda}; \mathcal{D}_c) + \gamma\|\boldsymbol{\Lambda}\|_1\right\}. \qquad \text{\# pre-train the network; } \hat{\boldsymbol{\Lambda}}_0 = \{\hat{\boldsymbol{\lambda}}_0^u, \hat{\boldsymbol{\lambda}}_0^v\}$$

6: **for** $k = 1, 2, \dots, n_{\max}$ **do**

7: Assemble the system states over the collocation points $\mathcal{D}_c^{\text{tr}}$ and $\mathcal{D}_c^{\text{va}}$:

$$\dot{\mathbf{U}}_u^{\text{tr}} = \bigcup_{i=1}^{N_c^{\text{tr}}} u_t(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{tr}}, t_i^{\text{tr}}) \quad \text{and} \quad \dot{\mathbf{U}}_u^{\text{va}} = \bigcup_{i=1}^{N_c^{\text{va}}} u_t(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{va}}, t_i^{\text{va}})$$
$$\dot{\mathbf{U}}_v^{\text{tr}} = \bigcup_{i=1}^{N_c^{\text{va}}} v_t(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{tr}}, t_i^{\text{tr}}) \quad \text{and} \quad \dot{\mathbf{U}}_v^{\text{va}} = \bigcup_{i=1}^{N_c^{\text{va}}} v_t(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{va}}, t_i^{\text{va}}).$$

8: Assemble the candidate library matrices over the collocation points $\mathcal{D}_c$, $\mathcal{D}_c^{\text{tr}}$ and $\mathcal{D}_c^{\text{va}}$:

$$\tilde{\boldsymbol{\Phi}} = \bigcup_{i=1}^{N_c} \boldsymbol{\phi}(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i, t_i), \quad \tilde{\boldsymbol{\Phi}}^{\text{tr}} = \bigcup_{i=1}^{N_c^{\text{tr}}} \boldsymbol{\phi}(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{tr}}, t_i^{\text{tr}}) \quad \text{and} \quad \tilde{\boldsymbol{\Phi}}^{\text{va}} = \bigcup_{i=1}^{N_c^{\text{va}}} \boldsymbol{\phi}(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{va}}, t_i^{\text{va}}).$$

9: Normalize candidate library matrices $\tilde{\boldsymbol{\Phi}}, \tilde{\boldsymbol{\Phi}}^{\text{tr}}$ and $\tilde{\boldsymbol{\Phi}}^{\text{va}}$ column-wisely ($j = 1, \dots, s$) to improve matrix condition:

$$\boldsymbol{\Phi}_{:,j} = \tilde{\boldsymbol{\Phi}}_{:,j} / \left\|\tilde{\boldsymbol{\Phi}}_{:,j}\right\|_2, \quad \boldsymbol{\Phi}_{:,j}^{\text{tr}} = \tilde{\boldsymbol{\Phi}}_{:,j}^{\text{tr}} / \left\|\tilde{\boldsymbol{\Phi}}_{:,j}^{\text{tr}}\right\|_2 \quad \text{and} \quad \boldsymbol{\Phi}_{:,j}^{\text{va}} = \tilde{\boldsymbol{\Phi}}_{:,j}^{\text{tr}} / \left\|\tilde{\boldsymbol{\Phi}}_{:,j}^{\text{tr}}\right\|_2.$$

Alternating Optimization Algorithm

Determine ℓ_0 regularization parameters $\beta^u = \kappa\mathcal{L}_p^u(\bar{\boldsymbol{\theta}}_0, \bar{\boldsymbol{\Lambda}}_0^u; \mathcal{D}_c^{va})$ and $\beta^v = \kappa\mathcal{L}_p^v(\bar{\boldsymbol{\theta}}_0, \bar{\boldsymbol{\Lambda}}_0^v; \mathcal{D}_c^{va})$. # κ can be determined via a Pareto front analysis, e.g., $\kappa = 1$.

Initialize the error indices:

$$\hat{\epsilon}^u = \mathcal{L}_p^u(\hat{\boldsymbol{\theta}}_{k-1}, \hat{\boldsymbol{\Lambda}}_{k-1}^u; \mathcal{D}_c^{va}) + \beta^u \|\hat{\boldsymbol{\Lambda}}_{k-1}^u\|_0 \text{ and } \hat{\epsilon}^v = \mathcal{L}_p^v(\hat{\boldsymbol{\theta}}_{k-1}, \hat{\boldsymbol{\Lambda}}_{k-1}^v; \mathcal{D}_c^{va}) + \beta^v \|\hat{\boldsymbol{\Lambda}}_{k-1}^v\|_0.$$

Set the initial threshold tolerance $\delta_1 = \Delta\delta$.

for $iter = 1, 2, ..., n_{\mathrm{str}}$ **do**

Run STRidge as shown in Algorithm 2 to determine:

$$\tilde{\boldsymbol{\Lambda}}^u = \texttt{STRidge}(\dot{\mathbf{U}}_u^{tr}, \boldsymbol{\Phi}^{tr}, \delta_{iter}) \quad \text{and} \quad \tilde{\boldsymbol{\Lambda}}^v = \texttt{STRidge}(\dot{\mathbf{U}}_v^{tr}, \boldsymbol{\Phi}^{tr}, \delta_{iter}).$$

Update the error indices:

$$\epsilon^u = \mathcal{L}_p^u(\hat{\boldsymbol{\theta}}_{k-1}, \tilde{\boldsymbol{\Lambda}}^u; \mathcal{D}_c^{va}) + \beta^u \|\tilde{\boldsymbol{\Lambda}}^u\|_0 \text{ and } \epsilon^v = \mathcal{L}_p^v(\hat{\boldsymbol{\theta}}_{k-1}, \tilde{\boldsymbol{\Lambda}}^v; \mathcal{D}_c^{va}) + \beta^v \|\tilde{\boldsymbol{\Lambda}}^v\|_0.$$

if $\epsilon^u \leqslant \hat{\epsilon}^u$ or $\epsilon^v \leqslant \hat{\epsilon}^v$ (run in parallel) **then**

Increase threshold tolerance with increment: $\delta_{iter+1} = \delta_{iter} + \Delta\delta$.

else

Decrease threshold tolerance increment $\Delta\delta = \Delta\delta/1.618$.

Update threshold tolerance with the new increment $\delta_{iter+1} = \max\{\delta_{iter} - 2\Delta\delta, 0\} + \Delta\delta$.

end if

end for

Return and re-scale the current best solution from STRidge cycles: $\hat{\boldsymbol{\Lambda}}_k = \{\tilde{\boldsymbol{\Lambda}}^u, \tilde{\boldsymbol{\Lambda}}^v\}$. # re-scaling due to normalization of $\boldsymbol{\Phi}$

Train the DNN via combined Adam and L-BFGS with $\{\mathcal{D}_u^{tr}, \mathcal{D}_c^{tr}\}$, and validate the trained model with $\{\mathcal{D}_u^{va}, \mathcal{D}_c^{va}\}$, namely,

$$\hat{\boldsymbol{\theta}}_k = \arg\min_{\boldsymbol{\theta}} \{\mathcal{L}_d(\boldsymbol{\theta}; \mathcal{D}_u) + \alpha\mathcal{L}_p(\boldsymbol{\theta}, \hat{\boldsymbol{\Lambda}}_k; \mathcal{D}_c)\}. \quad \text{\# train DNN given } \hat{\boldsymbol{\Lambda}}_k \text{ as known}$$

end for

Output: the best solution $\boldsymbol{\theta}_{\mathrm{best}} = \hat{\boldsymbol{\theta}}_{n_{\max}}$ and $\boldsymbol{\Lambda}_{\mathrm{best}} = \hat{\boldsymbol{\Lambda}}_{n_{\max}}$

Alternating Optimization Algorithm

Algorithm 2 Sequential threshold ridge regression (STRidge): $\hat{\boldsymbol{\lambda}} = \texttt{STRidge}(\dot{\mathbf{U}}, \boldsymbol{\Phi}, \delta)$

1: **Input:** Time derivative vector $\dot{\mathbf{U}}$, candidate function library matrix $\boldsymbol{\Phi}$, and threshold tolerance δ.
2: Inherit coefficients $\hat{\boldsymbol{\lambda}}$ from the DNN pre-training or the previous update.
3: **repeat**
4: Determine indices of coefficients in $\hat{\boldsymbol{\lambda}}$ falling below or above the sparsity threshold δ:

$$\mathcal{I} = \{i \in \mathcal{I} : |\hat{\lambda}_i| < \delta\} \text{ and } \mathcal{J} = \{j \in \mathcal{J} : |\hat{\lambda}_j| \geqslant \delta\}.$$

5: Enforce sparsity to small values by setting them to zero: $\hat{\boldsymbol{\lambda}}_{\mathcal{I}} = \mathbf{0}$.
6: Update remaining non-zero values with ridge regression:

$$\hat{\boldsymbol{\lambda}}_{\mathcal{J}} = \arg\min_{\boldsymbol{\lambda}_{\mathcal{J}}} \{\|\boldsymbol{\Phi}_{\mathcal{J}}\boldsymbol{\lambda}_{\mathcal{J}} - \dot{\mathbf{U}}\|_2^2 + 1 \times 10^{-5}\|\boldsymbol{\lambda}_{\mathcal{J}}\|_2^2\}. \quad \text{\# the parameter } 1 \times 10^{-5} \text{ is small and tunable}$$

7: **until** maximum number of iterations reached.
8: **Output:** The best solution $\hat{\boldsymbol{\lambda}} = \hat{\boldsymbol{\lambda}}_{\mathcal{I}} \cup \hat{\boldsymbol{\lambda}}_{\mathcal{J}}$

Results

Discovery of benchmark PDEs with single dataset

Burger's Equation →

Decaying stationary viscous shock of a system after a finite period of time, commonly found in simplified fluid mechanics, nonlinear acoustics and gas dynamics

$$u_t = -\boxed{u u_x} + \boxed{\nu u_{xx}}$$

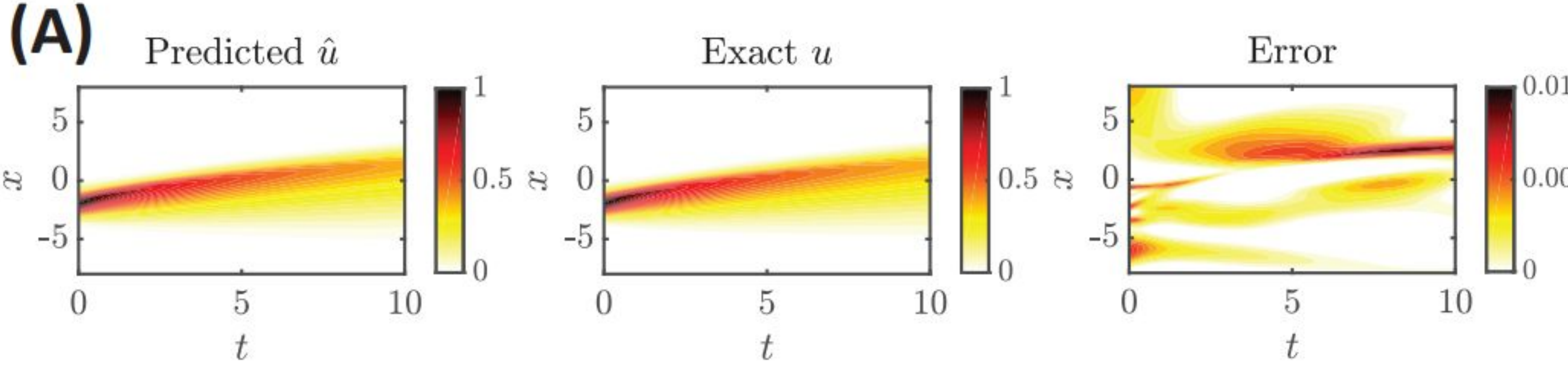
It represents a velocity field or wave amplitude, and the equation describes how this quantity evolves over time due to nonlinear advection and diffusion.

$$\phi \in \mathbb{R}^{1\times 16}$$

Discovery of benchmark PDEs with single dataset

Burger's Equation

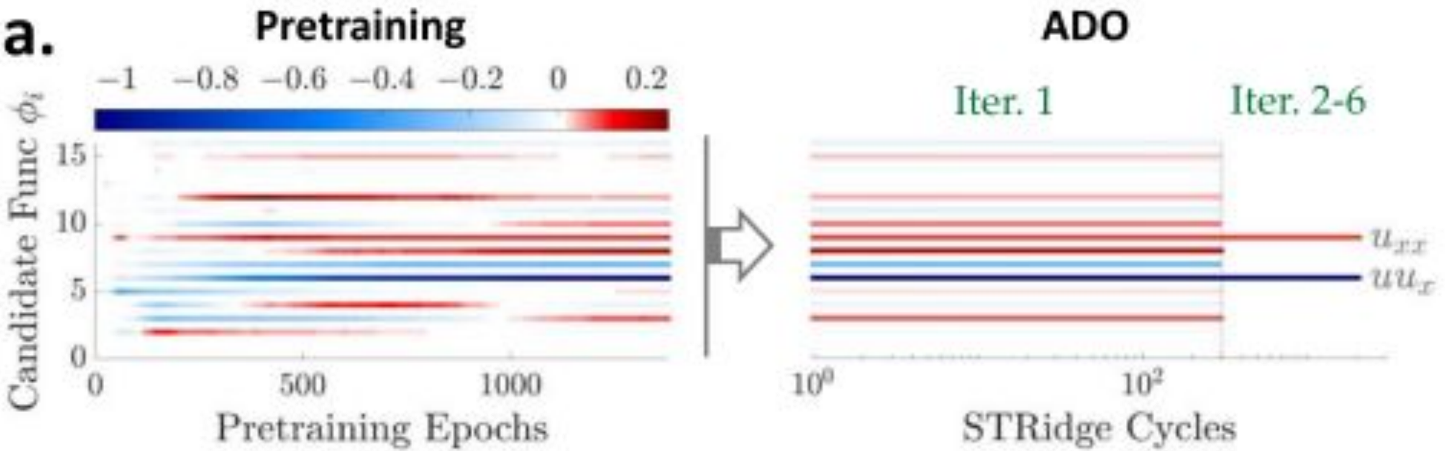
$$\phi \in \mathbb{R}^{1\times 16}$$



Discovery of benchmark PDEs with single dataset

Burger's Equation

$\phi \in \mathbb{R}^{1\times 16}$



Ground truth: $u_t = -uu_x + 0.1u_{xx}$

Discovered: $u_t = -1.009uu_x + 0.099u_{xx}$

Discovery of benchmark PDEs with single dataset

Kuramoto–Sivashinsky Equation → reaction-diffusion systems, flame front propagation, thin film dynamics, and turbulence in fluid systems.

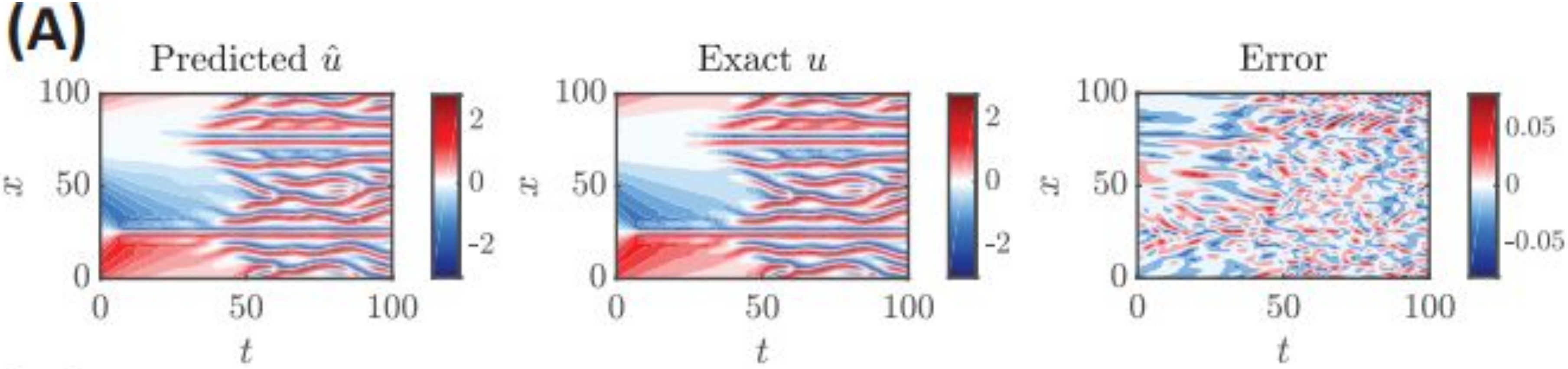
$$u_t = -uu_x - u_{xx} - u_{xxxx}$$

It can describe displacements, heights, velocities, or concentrations

$$\phi \in \mathbb{R}^{1\times 36}$$

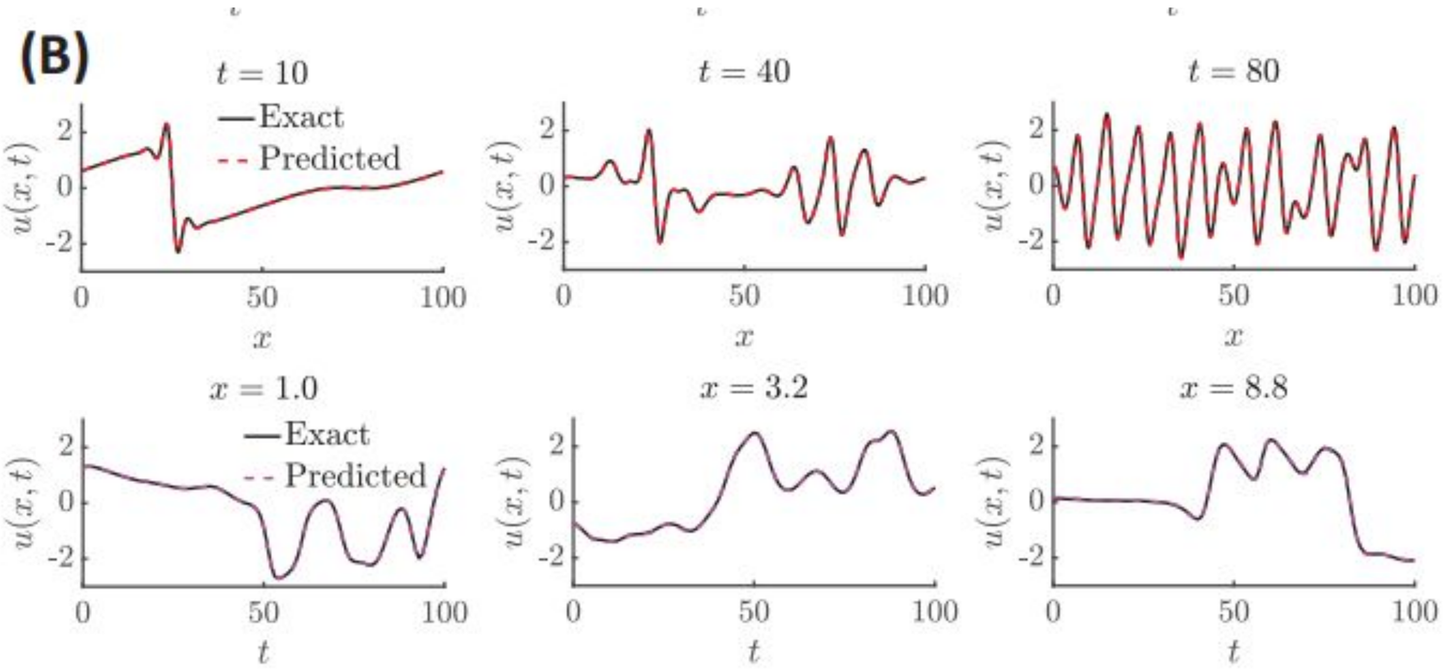
Discovery of benchmark PDEs with single dataset

Kuramoto–Sivashinsky Equation



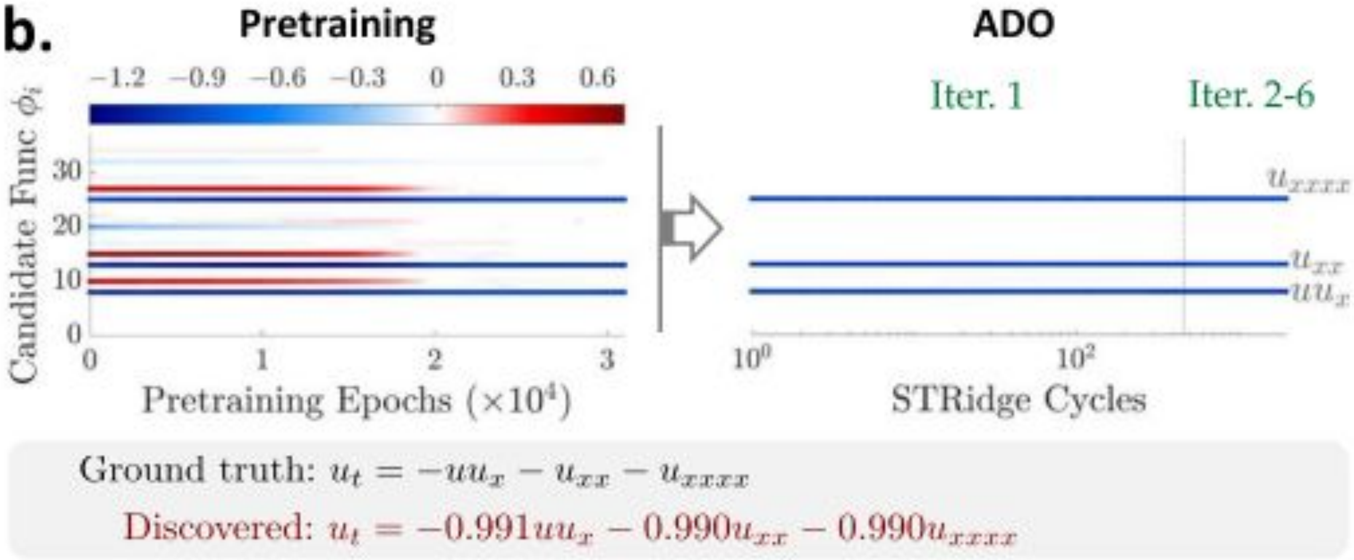
Discovery of benchmark PDEs with single dataset

Kuramoto–Sivashinsky Equation $\phi \in \mathbb{R}^{1\times 36}$



Discovery of benchmark PDEs with single dataset

Kuramoto–Sivashinsky Equation $\phi \in \mathbb{R}^{1\times 36}$



Discovery of benchmark PDEs with single dataset

No linear Schrodinger Equation

modeling the propagation of light in nonlinear optical fibers, Bose-Einstein condensates, Langmuir waves in hot plasmas

$$iu_t = -0.5u_{xx} - |u|^2 u$$

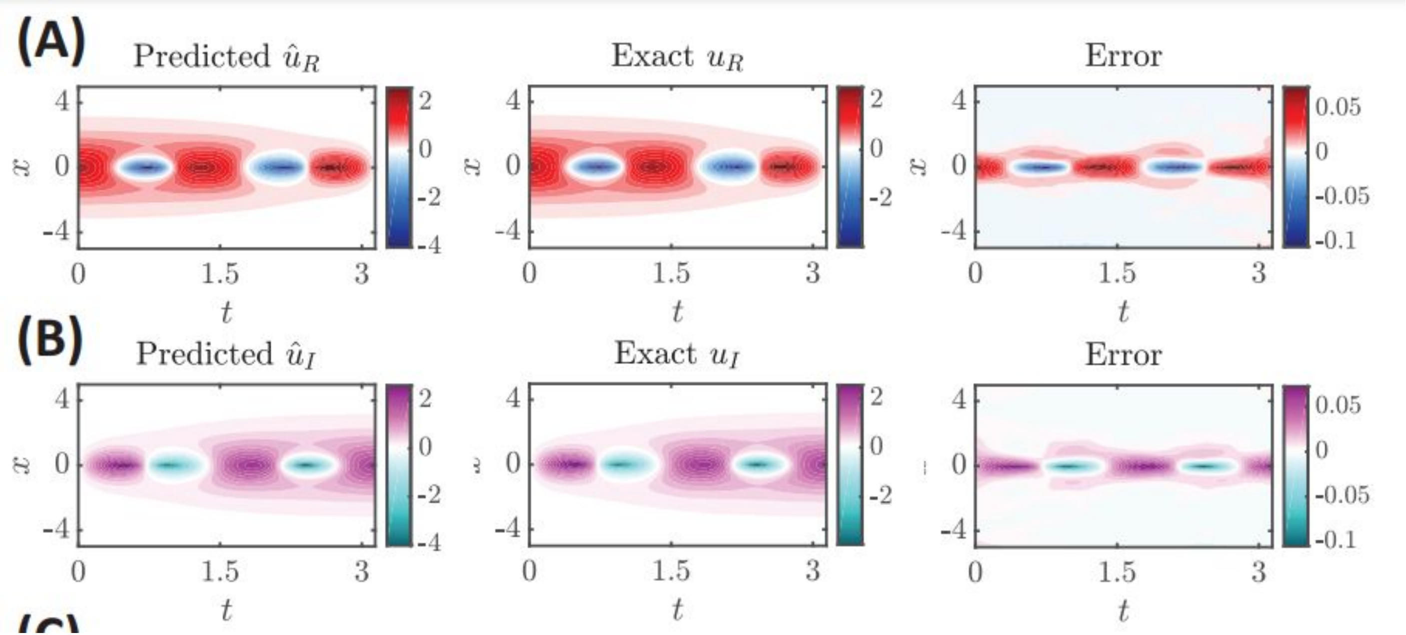
It describe the wave function. The nonlinear interaction term, which introduces self-interaction

$$\phi \in \mathbb{R}^{1\times 40}$$

Discovery of benchmark PDEs with single dataset

No linear Schrodinger Equation

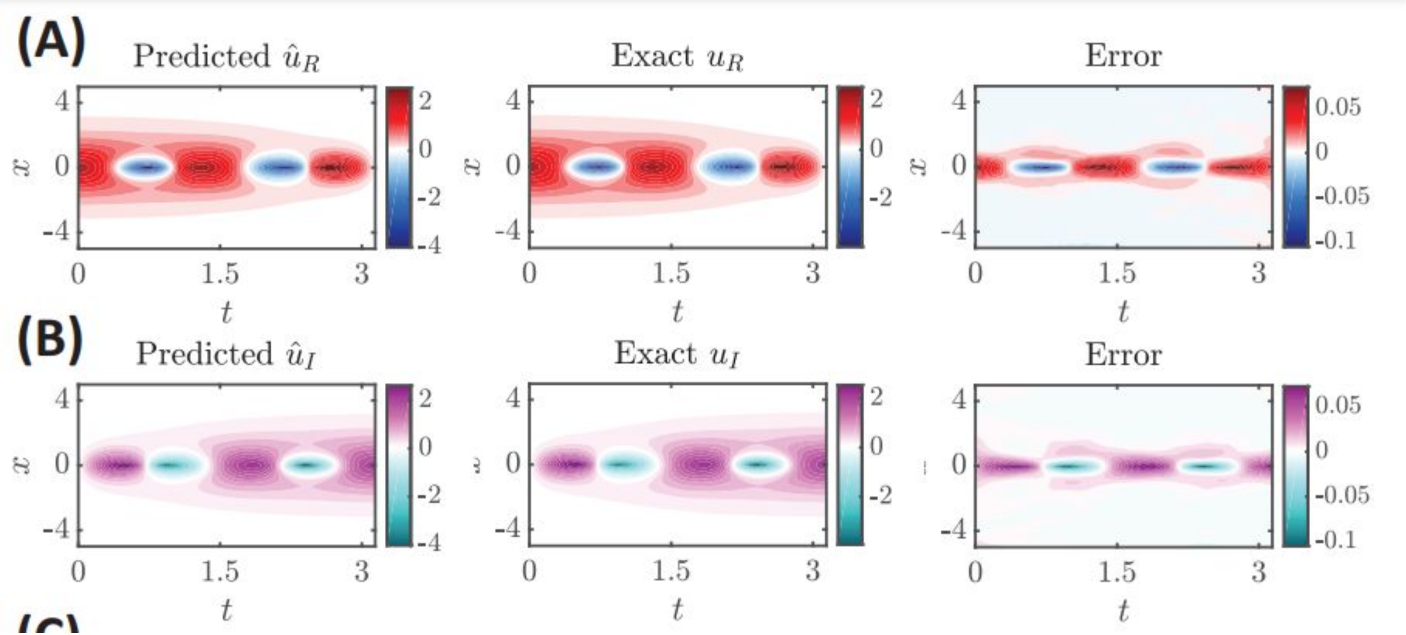
$$\phi \in \mathbb{R}^{1\times 40}$$



Discovery of benchmark PDEs with single dataset

No linear Schrodinger Equation

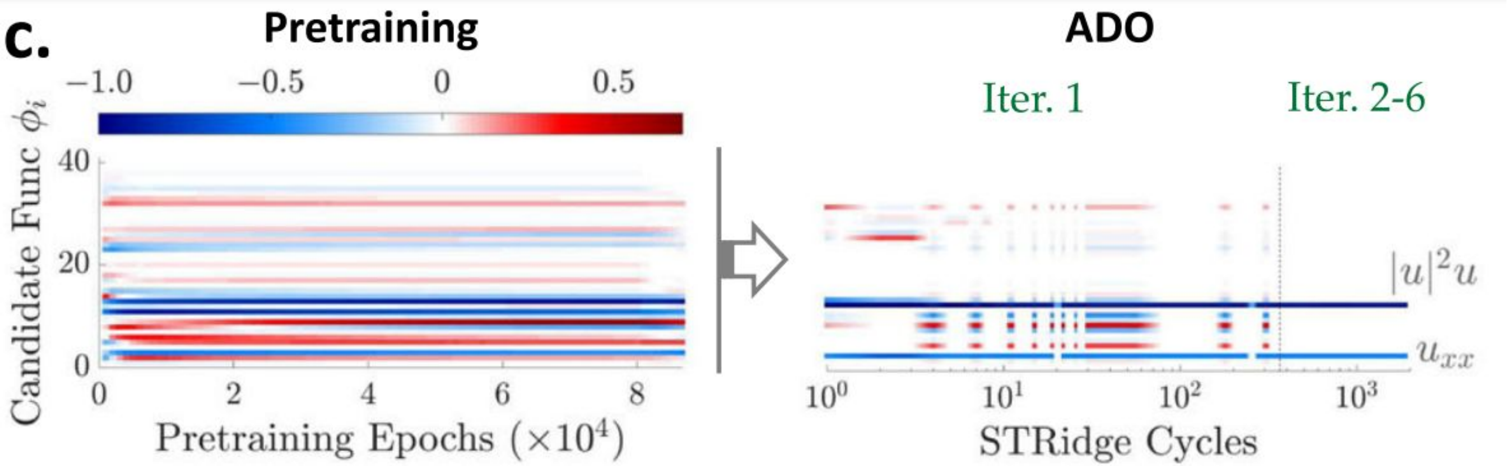
$$\phi \in \mathbb{R}^{1\times 40}$$



Discovery of benchmark PDEs with single dataset

No linear Schrodinger Equation

$$\phi \in \mathbb{R}^{1\times 40}$$



Ground truth: $iu_t = -0.5u_{xx} - |u|^2 u$

Discovered: $iu_t = -0.501u_{xx} - 1.000|u|^2 u$

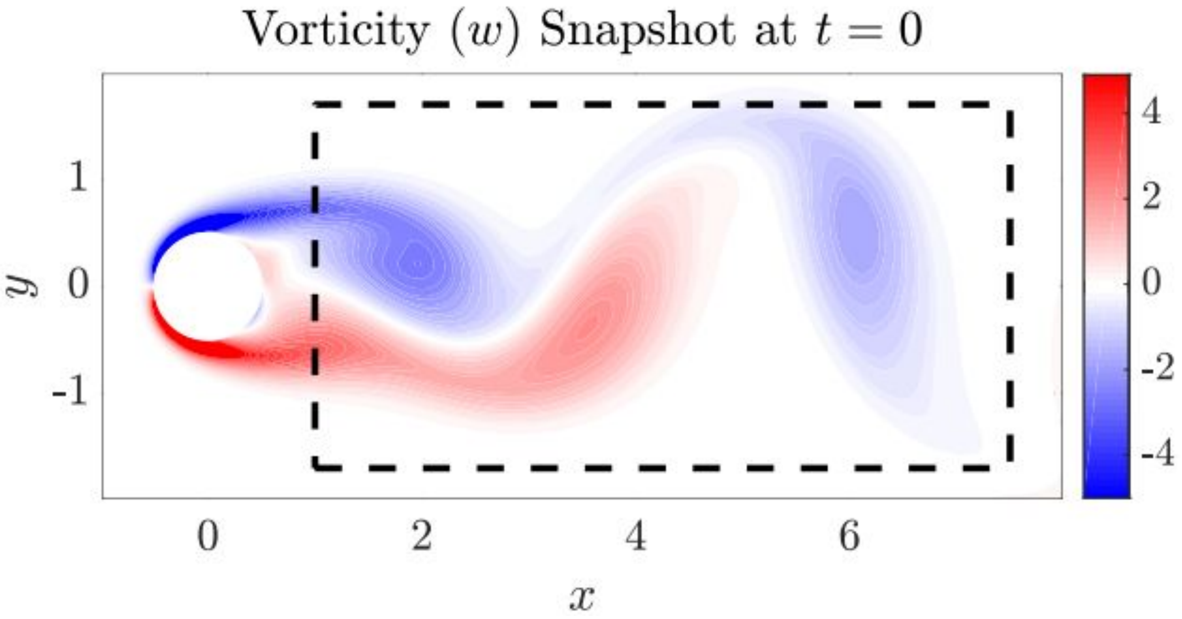
Discovery of benchmark PDEs with single dataset

Navier-Stokes Equation $\phi \in \mathbb{R}^{1\times 60}$

$$w_t = -uw_x - vw_y + 0.01w_{xx} + 0.01w_{yy}$$

Discovery of benchmark PDEs with single dataset

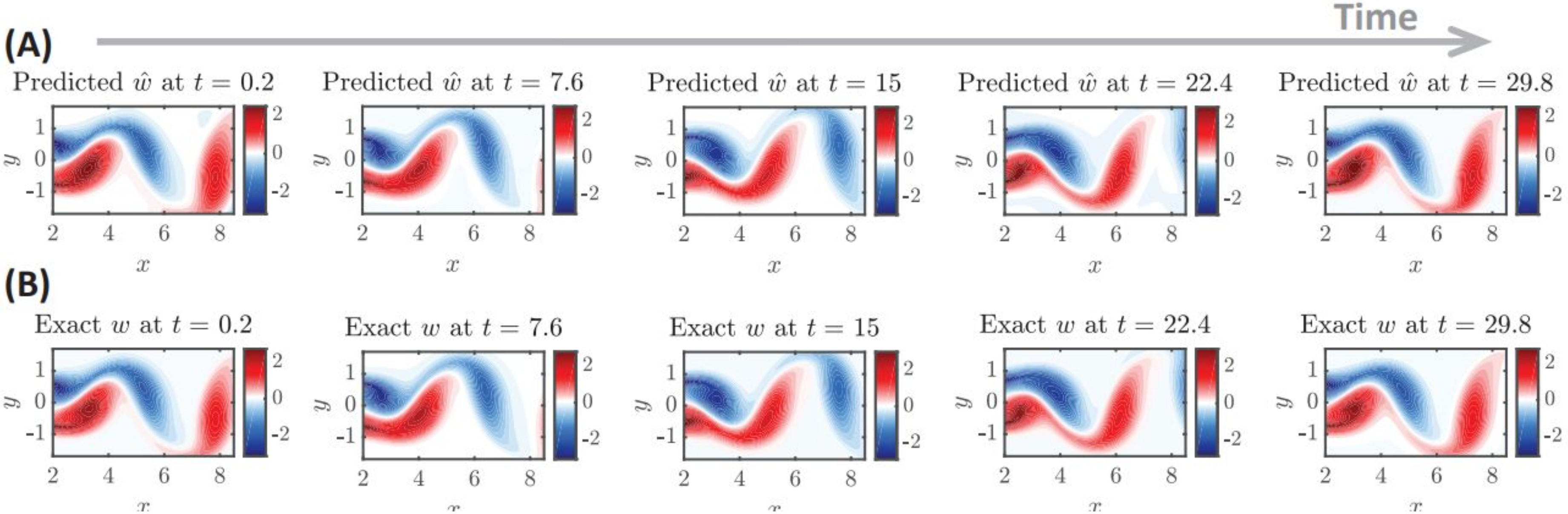
Navier-Stokes Equation $\phi \in \mathbb{R}^{1\times 60}$



Discovery of benchmark PDEs with single dataset

Navier-Stokes Equation

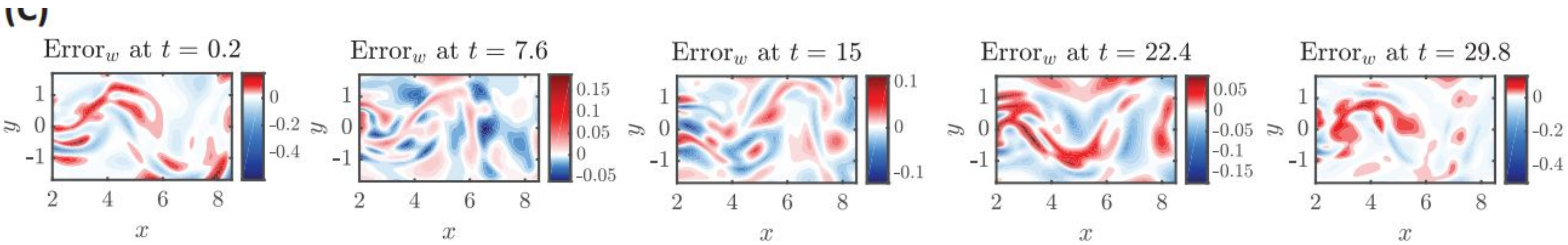
$$\phi \in \mathbb{R}^{1\times 60}$$



Discovery of benchmark PDEs with single dataset

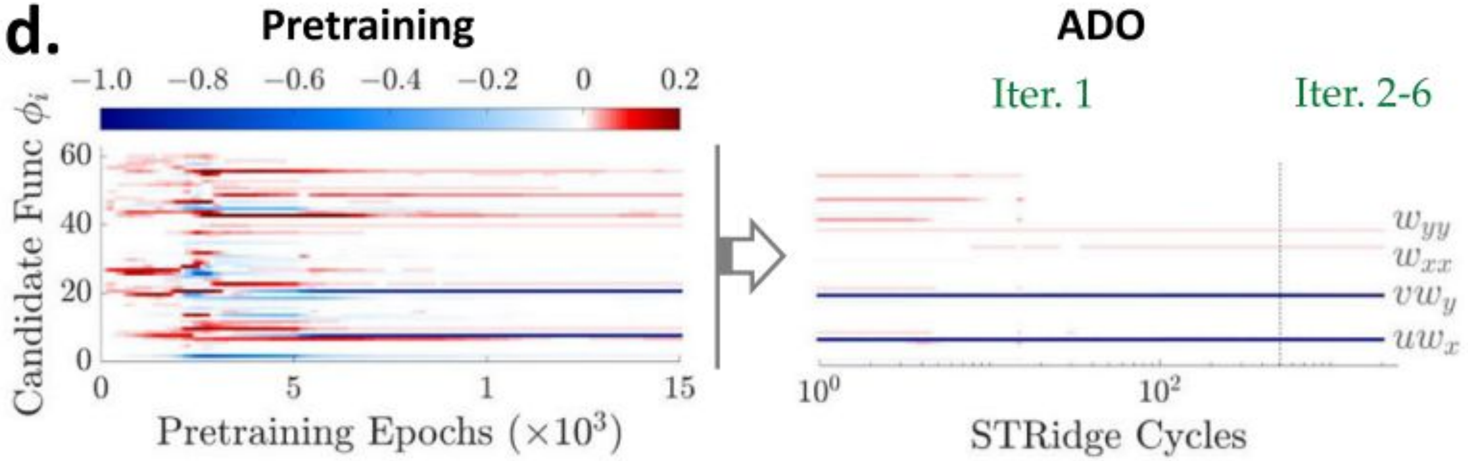
Navier-Stokes Equation

$$\phi \in \mathbb{R}^{1\times 60}$$



Discovery of benchmark PDEs with single dataset

Navier-Stokes Equation $\phi \in \mathbb{R}^{1\times 60}$



Ground truth: $w_t = 0.01 w_{xx} + 0.01 w_{yy} - u w_x - v w_y$

Discovered: $w_t = -0.996 u w_x - 0.991 v w_y + 0.010 w_{xx} + 0.010 w_{yy}$

Discovery of benchmark PDEs with single dataset

λ-ω reaction-diffusion equation

pattern formation and wave propagation in certain types of chemical and biological systems

$$u_t = 0.1\nabla^2 u + \lambda(g)u - \omega(g)v$$
$$v_t = 0.1\nabla^2 v + \omega(g)u + \lambda(g)v$$

Discovery of benchmark PDEs with single dataset

λ-ω reaction-diffusion equation → pattern formation and wave propagation in certain types of chemical and biological systems

$$u_t = \boxed{0.1\nabla^2 u} + \boxed{\lambda(g)u - \omega(g)v}$$
$$v_t = \boxed{0.1\nabla^2 v} + \boxed{\omega(g)u + \lambda(g)v}$$

diffusion terms

reaction terms

$$\phi \in \mathbb{R}^{1\times 110}$$

Discovery of benchmark PDEs with single dataset

λ-ω reaction-diffusion equation

pattern formation and wave propagation in certain types of chemical and biological systems

$$u_t = \boxed{0.1\nabla^2 u} + \boxed{\lambda(g)u - \omega(g)v}$$
$$v_t = \boxed{0.1\nabla^2 v} + \boxed{\omega(g)u + \lambda(g)v}$$
$$g = u^2 + v^2$$
$$\omega = -g^2$$
$$\lambda = 1 - g^2$$

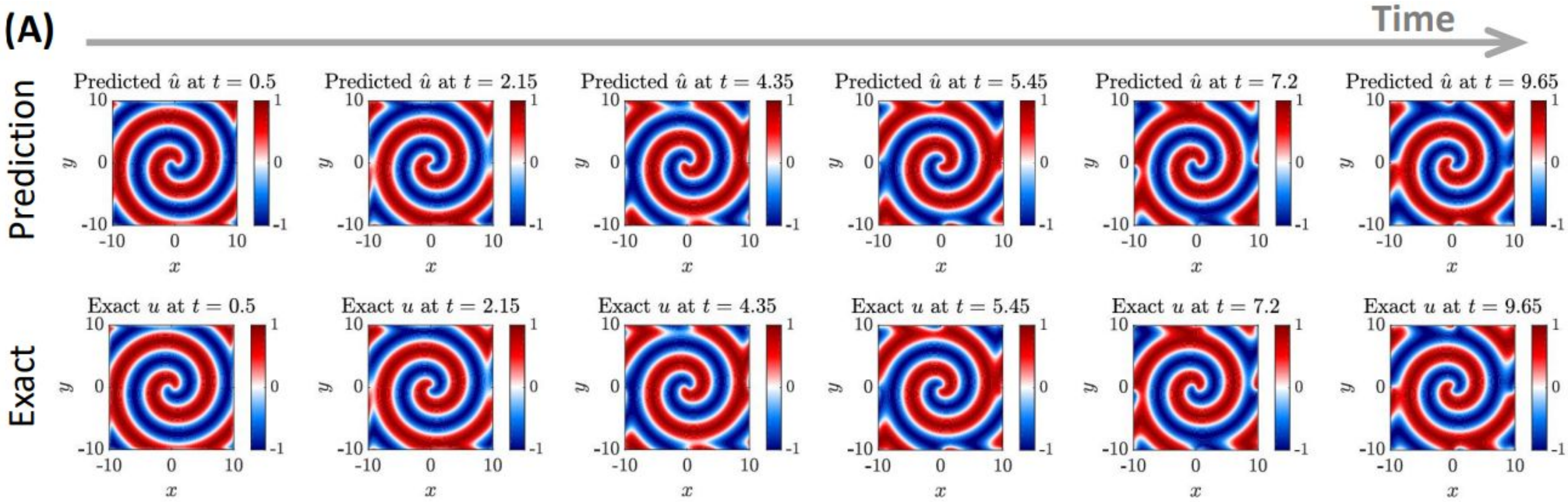
$$\phi \in \mathbb{R}^{1\times 110}$$

diffusion terms

reaction terms

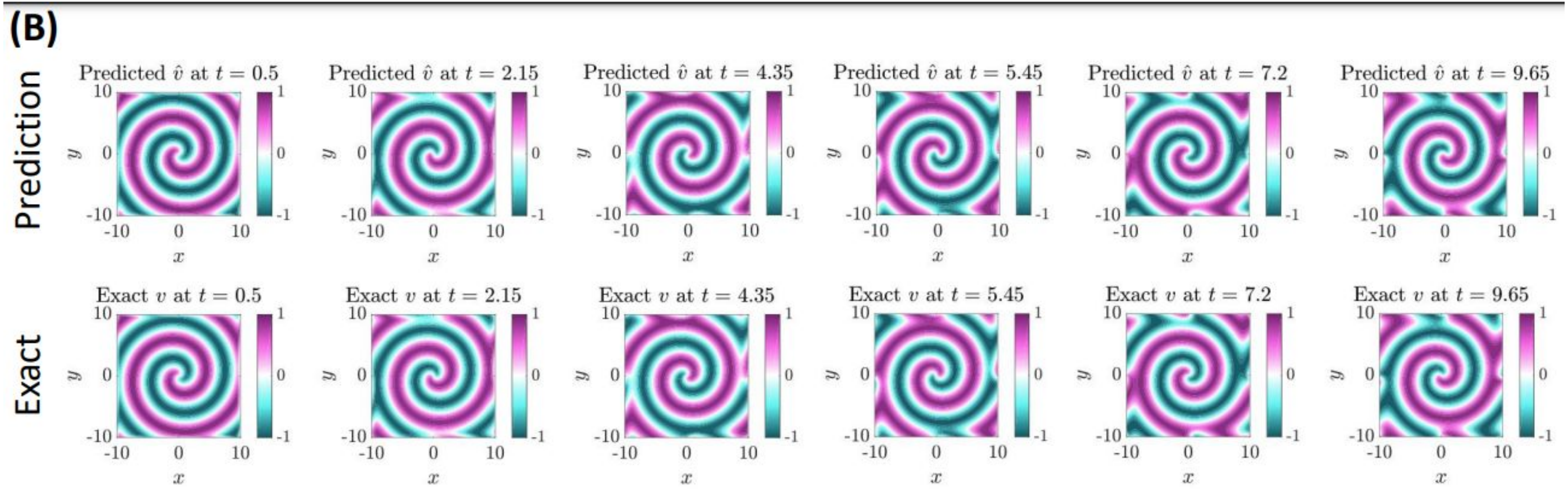
Discovery of benchmark PDEs with single dataset

λ-ω reaction-diffusion equation $\phi \in \mathbb{R}^{1\times 110}$



Discovery of benchmark PDEs with single dataset

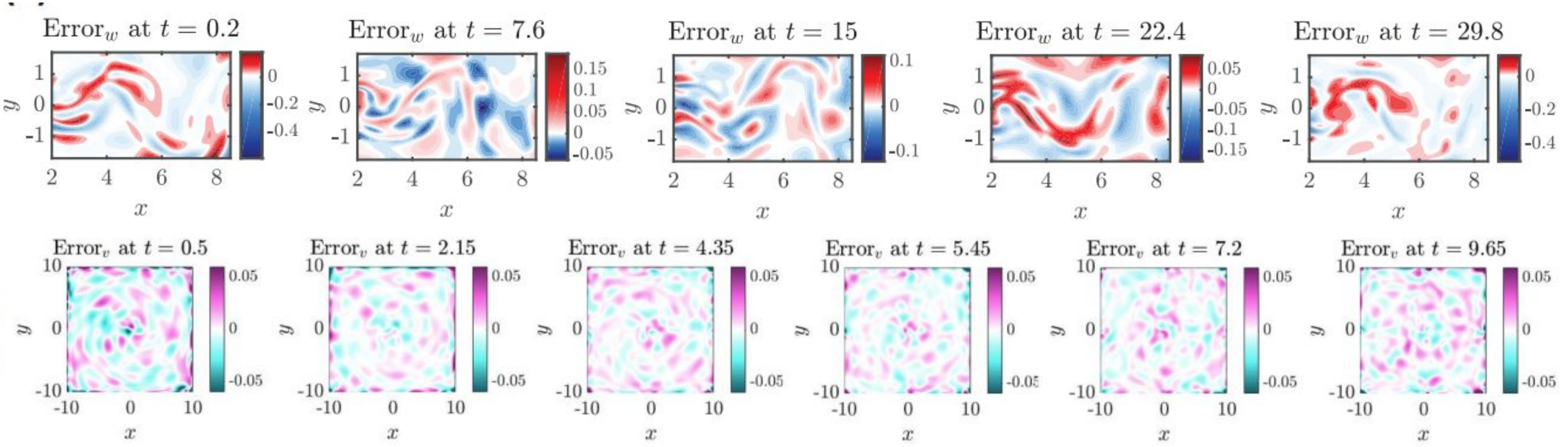
λ-ω reaction-diffusion equation $\phi \in \mathbb{R}^{1\times 110}$



Discovery of benchmark PDEs with single dataset

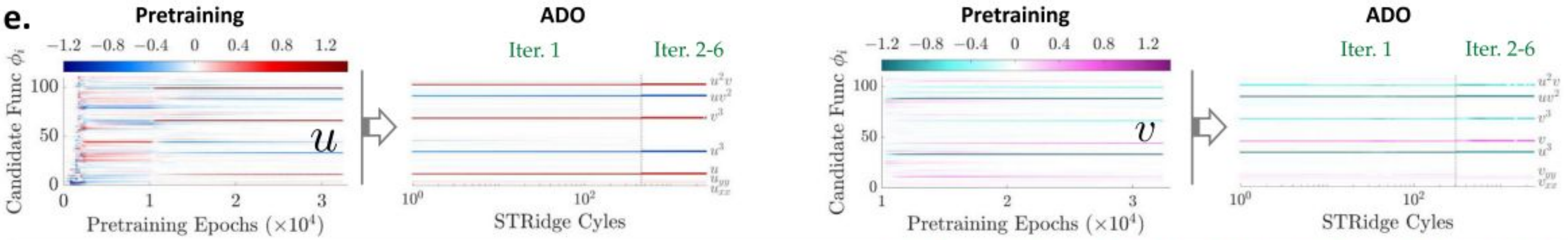
λ-ω reaction-diffusion equation

$$\phi \in \mathbb{R}^{1\times 110}$$



Discovery of benchmark PDEs with single dataset

λ-ω reaction-diffusion equation $\phi \in \mathbb{R}^{1\times 110}$



Ground truth:
$$u_t = 0.1u_{xx} + 0.1u_{yy} - uv^2 - u^3 + v^3 + u^2v + u$$
$$v_t = 0.1v_{xx} + 0.1v_{yy} - uv^2 - u^3 - v^3 - u^2v + v$$

Discovered:
$$u_t = 0.097u_{xx} + 0.097u_{yy} - 0.955uv^2 - 0.964u^3 + 0.997v^3 + 0.997u^2v + 0.964u$$
$$v_t = 0.099v_{xx} + 0.101v_{yy} - 1.009uv^2 - 1.002u^3 - 0.982v^3 - 0.989u^2v + 0.982v$$

Summary

Table 1 Summary of the PINN-SR discovery results in the context of accuracy for a range of canonical models.

PDE name	Err. (N-0%)	Err. (N-1%)	Err. (N-10%)	Description of data discretization
Burgers'	0.01 ± 0.01%	0.19 ± 0.11%	0.88 ± 0.03%	$x \in [-8, 8]_{\tilde{n}=256}$, $t \in [0, 10]_{\tilde{n}=101}$, sub. 3.19%
KS	0.07 ± 0.01%	0.61 ± 0.04%	0.94 ± 0.05%	$x \in [0, 100]_{\tilde{n}=1024}$, $t \in [0, 100]_{\tilde{n}=251}$, sub. 12.6%%
Schrödinger	0.09 ± 0.04%	0.65 ± 0.29%	0.08 ± 0.03%	$x \in [-4.5, 4.5]_{\tilde{n}=512}$, $t \in [0, \pi]_{\tilde{n}=501}$, sub. 37.5%
NS	0.66 ± 0.72%	0.86 ± 0.63%	1.22 ± 0.69%	$x \in [0, 9]_{\tilde{n}=449}$, $y \in [-2, 2]_{\tilde{n}=199}$, $t \in [0, 30]_{\tilde{n}=151}$, sub. 0.22%
λ-ω RD	0.07 ± 0.08%	0.25 ± 0.30%	1.84 ± 1.48%	$x, y \in [-10, 10]_{\tilde{n}=256}$, $t \in [0, 10]_{\tilde{n}=201}$, sub. 0.29%

The error is defined as the average relative error of the identified non-zero coefficients w.r.t. the ground truth. The percentage values in the parentheses denote the noise levels (e.g., noise free 0%, 1% and 10%) and the subscript $\tilde{n}$ represents the number of discretization. Our method is also compared with SINDy (the PDE-FIND approach presented in ref. [6]) as illustrated in Supplementary Table 1. It is noted that much less measurement data polluted with a higher level of noise are used in our discovery. Gaussian white noise is added to the synthetic response with the noise level defined as the root-mean-square ratio between the noise and the exact solution.

Comparison with SINDy

PDE name	Method	Error (noise 0%)	Error (noise 1%)	Error (noise 10%)	# of Measurement points
Burgers'	PINN-SR	0.01±0.01%	0.19±0.11%	0.88 ± 0.03%	~1k
	PDE-FIND	Fail 0.15±0.06%	Fail 0.80±0.60%	Fail Fail	~1k ~26k
KS	PINN-SR	0.07±0.01%	0.61±0.04%	0.94 ± 0.05%	~32k
	PDE-FIND	35.75±16.30% 1.30±1.30%	Fail 52.00±1.40%	Fail Fail	~32k ~257k
Schrödinger	PINN-SR	0.09±0.04%	0.65±0.29%	0.08±0.03%	~96k
	PDE-FIND	Fail 0.05±0.01%	Fail 3.00±1.00%	Fail Fail	~96K ~257k
NS	PINN-SR	0.66±0.72%	0.86±0.63%	1.22±0.69%	~30k
	PDE-FIND	Fail 1.00±0.20%	Fail 7.00±6.00%	Fail Fail	~30K ~300k
λ-ω RD	PINN-SR	0.07±0.08%	0.25±0.30%	1.84 ± 1.48%	~37.5k
	PDE-FIND	Fail 0.02±0.02%	Fail Fail	Fail Fail	~37.5k ~150k

Discovery of PDEs with multiple independent datasets

$$\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}$$

Discovery of PDEs with multiple independent datasets

$$\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}$$

$$\mathcal{I}[\mathbf{x} \in \Omega, t = 0; \mathbf{u}; \mathbf{u}_t] = \mathbf{g}_1(\mathbf{x})$$
$$\mathcal{B}[\mathbf{x} \in \partial\Omega; \mathbf{u}; \nabla_{\mathbf{x}} \mathbf{u}] = \mathbf{h}_1(t)$$

Discovery of PDEs with multiple independent datasets

$$\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}$$

$$\mathcal{I}[\mathbf{x} \in \Omega, t = 0; \mathbf{u}; \mathbf{u}_t] = \mathbf{g}_1(\mathbf{x})$$
$$\mathcal{B}[\mathbf{x} \in \partial\Omega; \mathbf{u}; \nabla_{\mathbf{x}} \mathbf{u}] = \mathbf{h}_1(t)$$

$$\mathcal{I}[\mathbf{x} \in \Omega, t = 0; \mathbf{u}; \mathbf{u}_t] = \mathbf{g}_2(\mathbf{x})$$
$$\mathcal{B}[\mathbf{x} \in \partial\Omega; \mathbf{u}; \nabla_{\mathbf{x}} \mathbf{u}] = \mathbf{h}_2(t)$$

Discovery of PDEs with multiple independent datasets

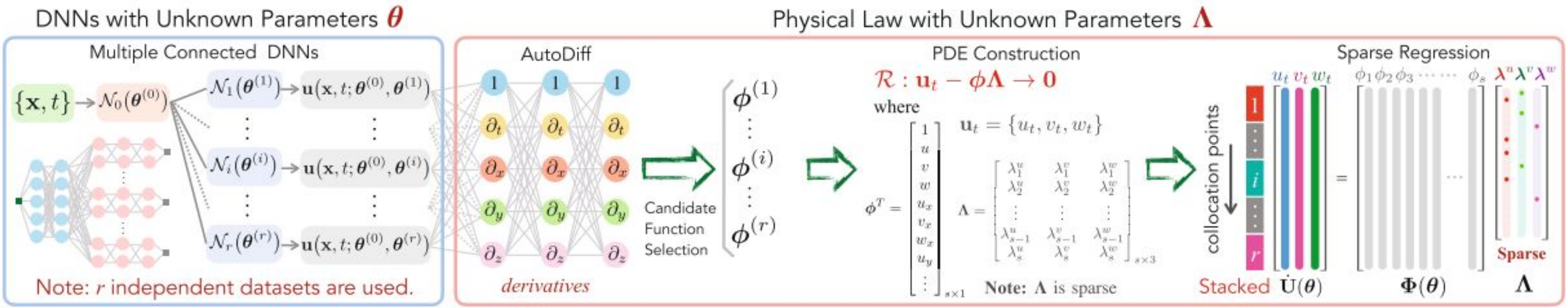
$$\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}$$

$$\mathcal{I}[\mathbf{x} \in \Omega, t = 0; \mathbf{u}; \mathbf{u}_t] = \mathbf{g}_1(\mathbf{x})$$
$$\mathcal{B}[\mathbf{x} \in \partial\Omega; \mathbf{u}; \nabla_{\mathbf{x}} \mathbf{u}] = \mathbf{h}_1(t)$$

$$\mathcal{I}[\mathbf{x} \in \Omega, t = 0; \mathbf{u}; \mathbf{u}_t] = \mathbf{g}_2(\mathbf{x})$$
$$\mathcal{B}[\mathbf{x} \in \partial\Omega; \mathbf{u}; \nabla_{\mathbf{x}} \mathbf{u}] = \mathbf{h}_2(t)$$

$$\mathcal{I}[\mathbf{x} \in \Omega, t = 0; \mathbf{u}; \mathbf{u}_t] = \mathbf{g}_r(\mathbf{x})$$
$$\mathcal{B}[\mathbf{x} \in \partial\Omega; \mathbf{u}; \nabla_{\mathbf{x}} \mathbf{u}] = \mathbf{h}_r(t)$$

Discovery of PDEs with multiple independent datasets



Discovery of PDEs with multiple independent datasets

Burger's Equation $u_t = -uu_x + 0.0032u_{xx}$ $\phi \in \mathbb{R}^{1\times 16}$

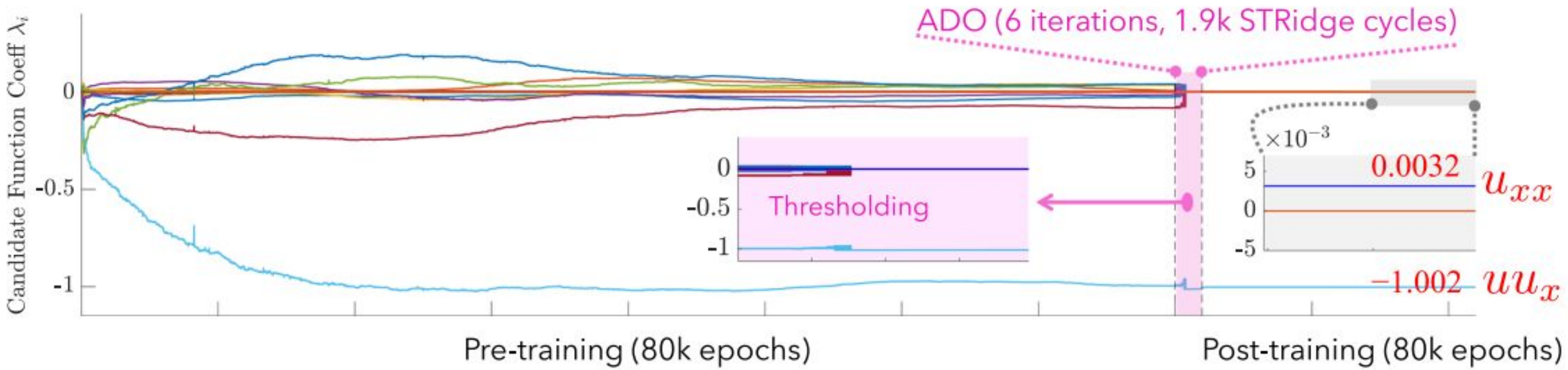
I/BC 1: $u(x,0) = -\sin(\pi x), u(-1,t) = u(1,t) = 0$

I/BC 2: $u(x,0) = \mathcal{G}(x), u(-1,t) = u(1,t) = 0$

I/BC 3: $u(x,0) = -x^3, u(-1,t) = 1, u(1,t) = -1$

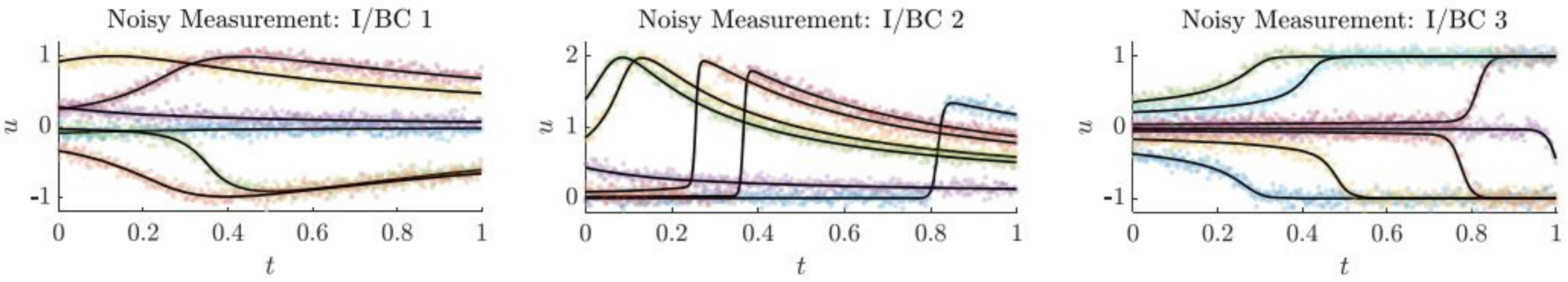
Discovery of PDEs with multiple independent datasets

Burger's Equation $u_t = -uu_x + 0.0032u_{xx}$ $\phi \in \mathbb{R}^{1\times 16}$

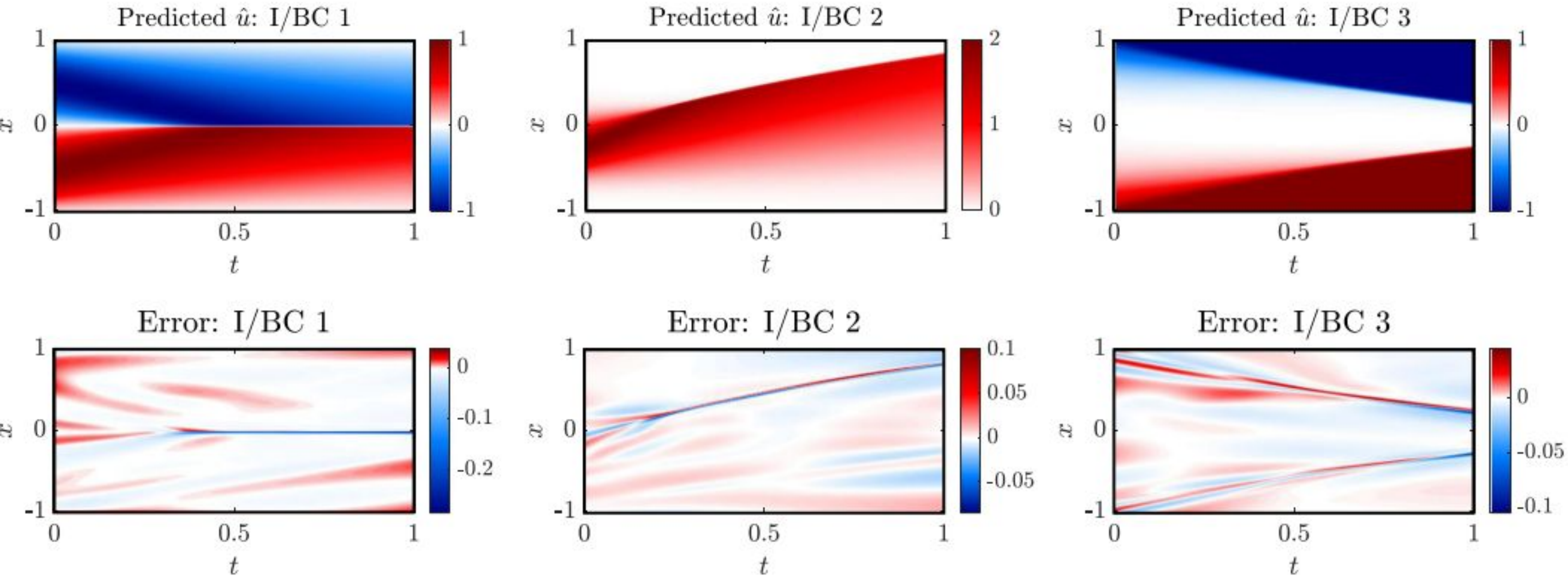


Discovery of PDEs with multiple independent datasets

Burger's Equation $u_t = -uu_x + 0.0032u_{xx}$ $\phi \in \mathbb{R}^{1\times 16}$



Discovery of PDEs with multiple independent datasets



Ground truth: $u_t + uu_x - 0.0032u_{xx} = 0$

Discovered: $u_t + 1.002uu_x - 0.0032u_{xx} = 0$

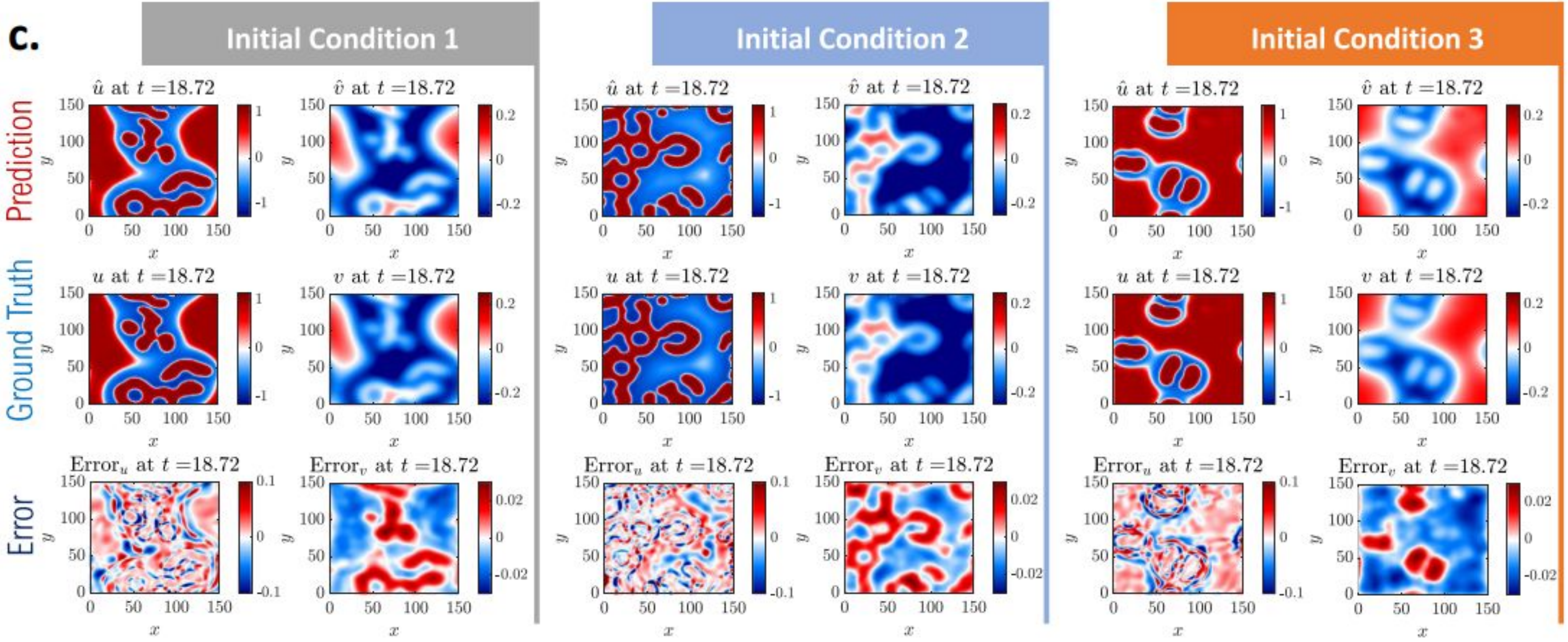
Discovery of PDEs with multiple independent datasets

FitzHugh–Nagumo (FN) reaction–diffusion system Equation

$$u_t = \gamma_u \Delta u + u - u^3 - v + \alpha$$
$$v_t = \gamma_v \Delta v + \beta(u - v).$$

FN equations are commonly used to describe biological neuron activities excited by external stimulus (α), which exhibit an activator-inhibitor system because one equation boosts the production of both components while the other equation dissipates their new growth.

Discovery of PDEs with multiple independent datasets



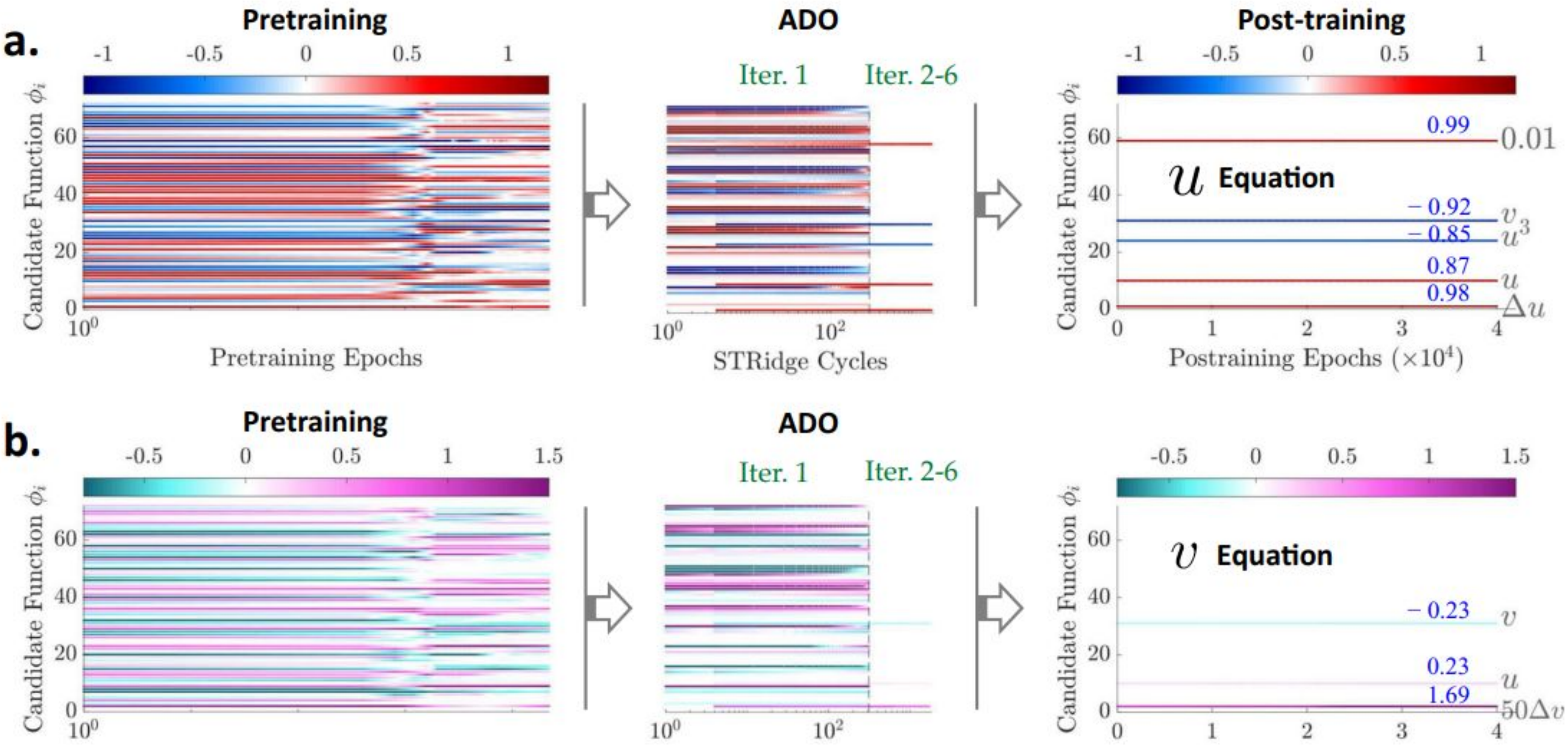
Ground truth: $u_t = \Delta u + u - u^3 - v + 0.01$

$v_t = 100\Delta v + 0.25u - 0.25v$

Discovered: $u_t = 0.975\Delta u + 0.871u - 0.847u^3 - 0.924v + 0.010$

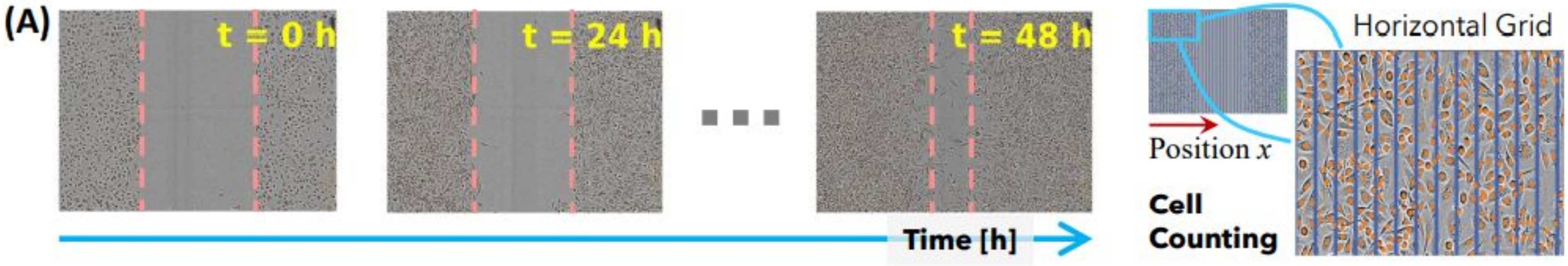
$v_t = 84.339\Delta v + 0.225u - 0.229v$

Discovery of PDEs with multiple independent datasets

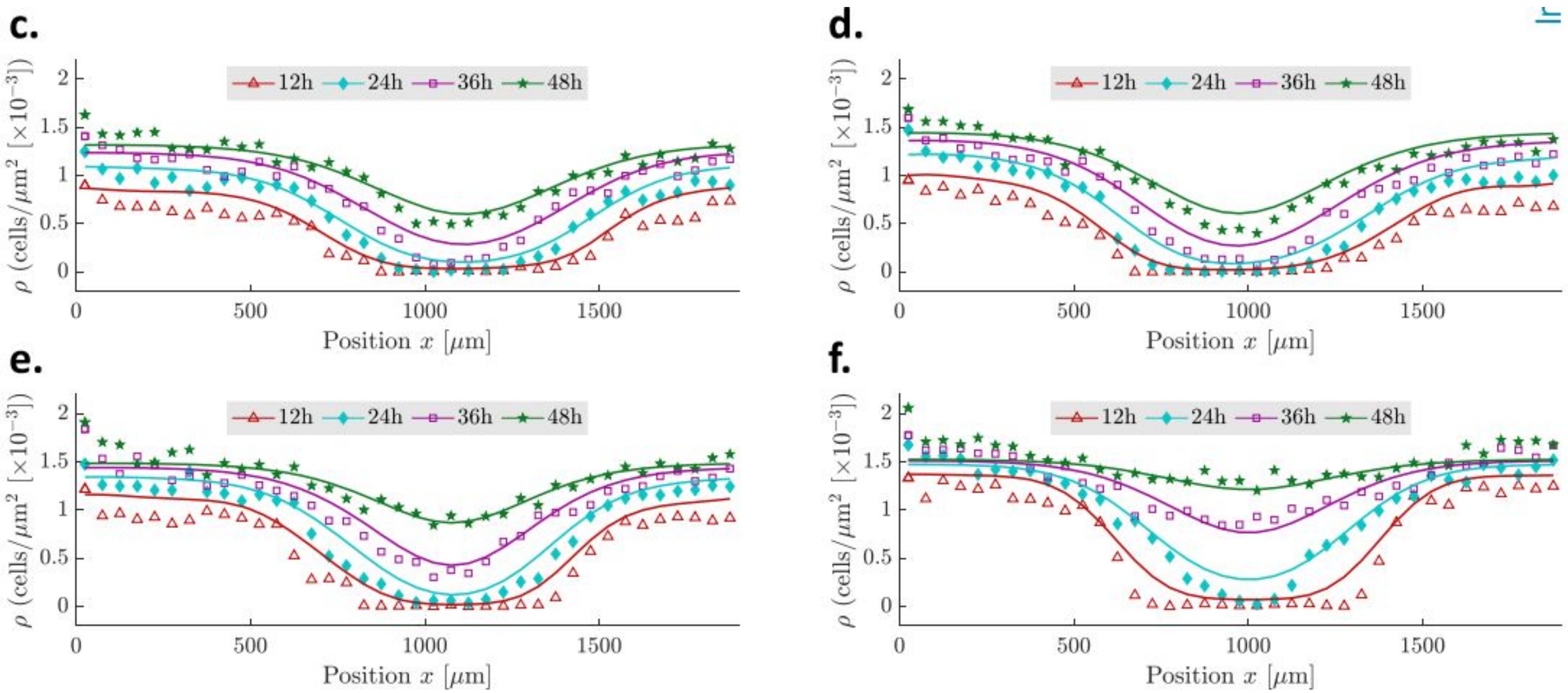


Probes in vitro

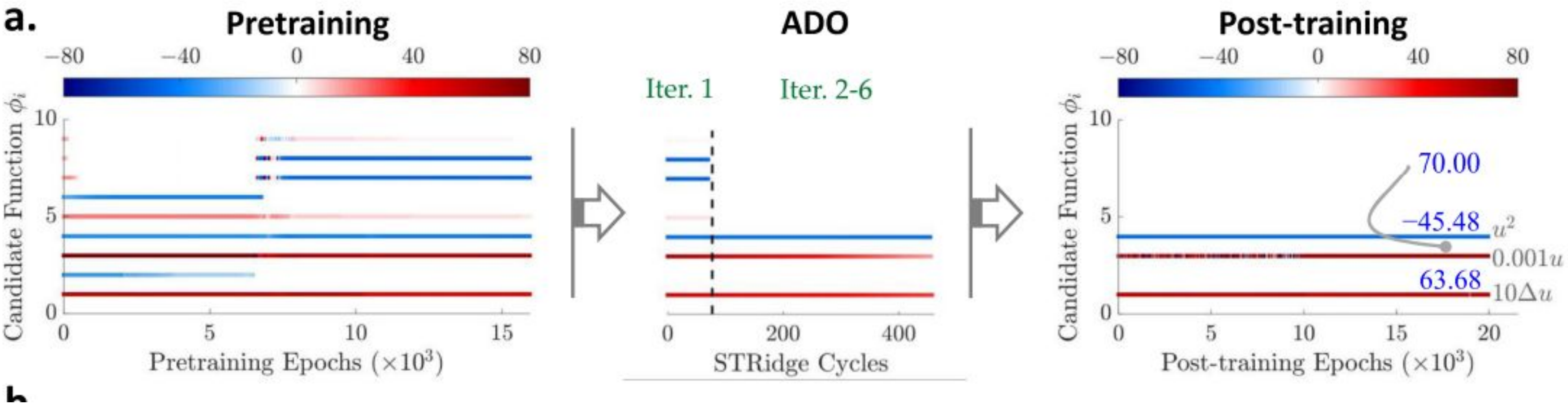
Experimental discovery of cell migration and proliferation



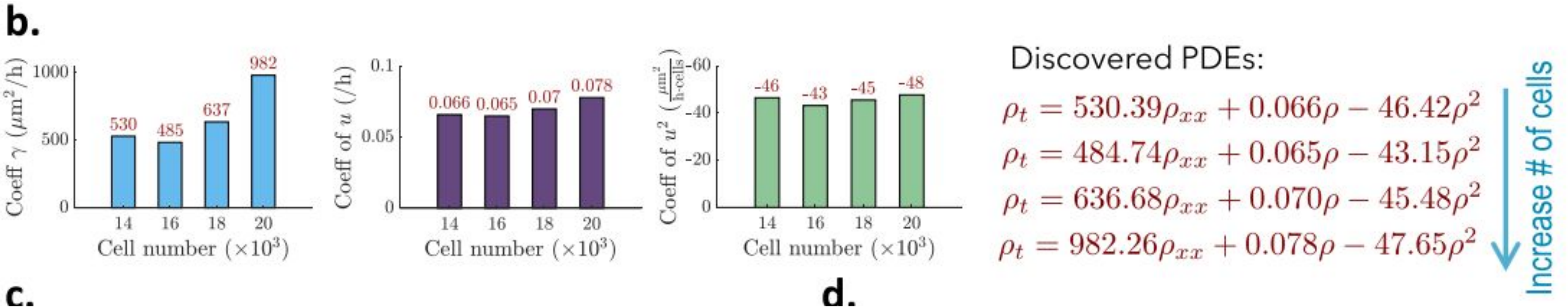
Experimental discovery of cell migration and proliferation



Experimental discovery of cell migration and proliferation



Experimental discovery of cell migration and proliferation



$$\rho_t = \gamma\rho_{xx} + \lambda_1\rho + \lambda_2\rho^2$$

Fisher-Kolmogorov model

Discussion

Discussion

- **Advantages of DNNs**: handling noise and scarce data effectively using collocation points that are not tied to measurements.

Discussion

- **Advantages of DNNs**: handling noise and scarce data effectively using collocation points that are not tied to measurements.
- Handling Multiple Datasets.

Discussion

- **Advantages of DNNs**: handling noise and scarce data effectively using collocation points that are not tied to measurements.
- Handling Multiple Datasets.
- Alternating Direction Optimization: The framework optimizes both DNN training and the selection of sparse coefficients to reconstruct governing PDEs simultaneously.

Discussion

- **Advantages of DNNs**: handling noise and scarce data effectively using collocation points that are not tied to measurements.
- Handling Multiple Datasets.
- Alternating Direction Optimization: The framework optimizes both DNN training and the selection of sparse coefficients to reconstruct governing PDEs simultaneously.
- Robustness: The method demonstrates resilience to both Gaussian and non-Gaussian noise and can accurately identify governing equations from sparse, noisy data.

Limitations

- Scalability issues with the "root-branch" scheme when dealing with multiple independent datasets.
- Inapplicability to systems where PDE coefficients vary over time or space (although future extensions are possible).
- Difficulty modeling chaotic behaviors or sharp propagating wavefronts due to the global basis approach.
- Dependency on a pre-defined library of candidate terms for PDE discovery, which can be hard to design.

Thanks!

Any questions?

Possibles questions and answers

If there is any source ?

$$\boxed{\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}}$$

$$\boxed{\mathbf{p} = \mathbf{p}(\mathbf{x}, t)}$$

If there is any source ?

$$\boxed{\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots; \lambda\right] = \mathbf{p}}$$

$$\boxed{\mathbf{u}_t = [\phi^u \;\; \phi^p][\boldsymbol{\Lambda}^u \;\; \boldsymbol{\Lambda}^p]^T}$$

If there is any source ?

$$u_t + uu_x - 0.1u_{xx} = \sin(x)\sin(t)$$

If there is any source ?

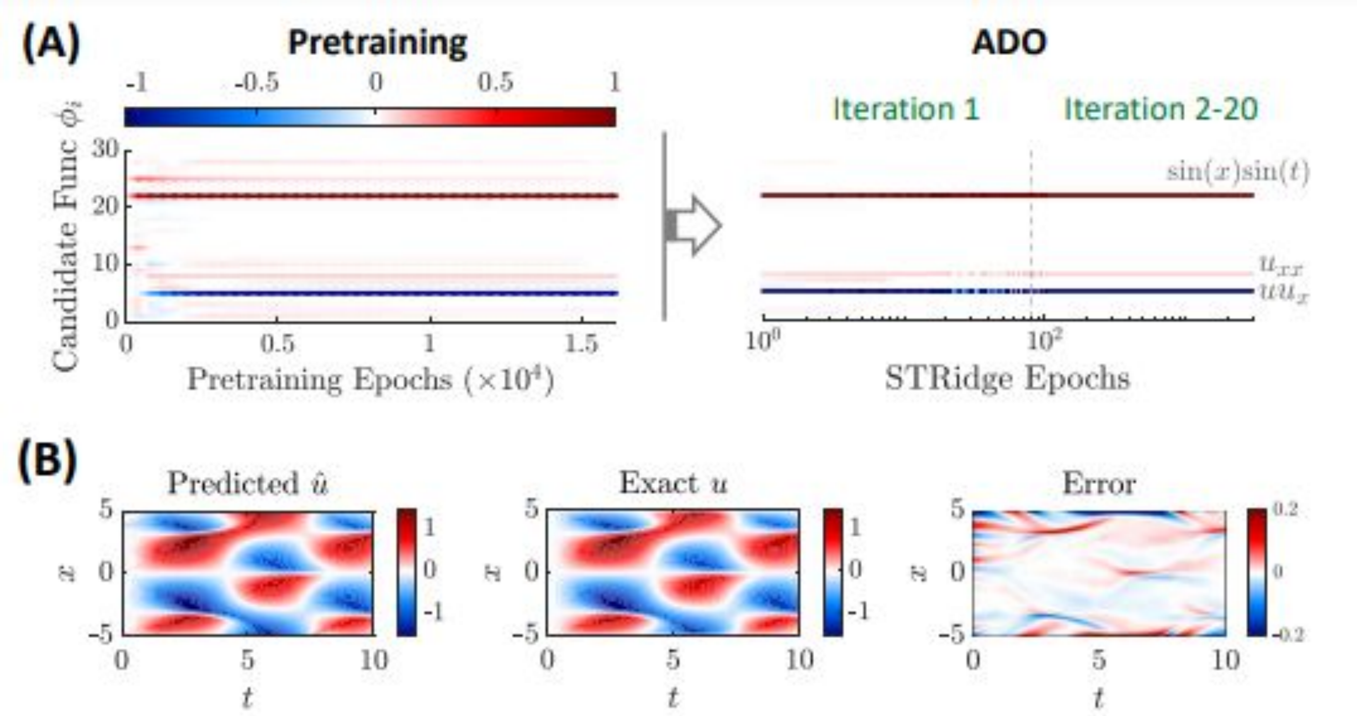
$$u_t + uu_x - 0.1u_{xx} = \sin(x)\sin(t)$$

$$\phi^u = \{1, u, u^2, u^3, u_x, uu_x, u^2u_x, u^3u_x, u_{xx}, uu_{xx}, u^2u_{xx}, u^3u_{xx}, u_{xxx}, uu_{xxx}, u^2u_{xxx}, u^3u_{xxx}\}$$

$$\phi^p = \{a, b, c, d, a^2, b^2, c^2, d^2, ac, ab, ad, bc, bd, cd\}$$

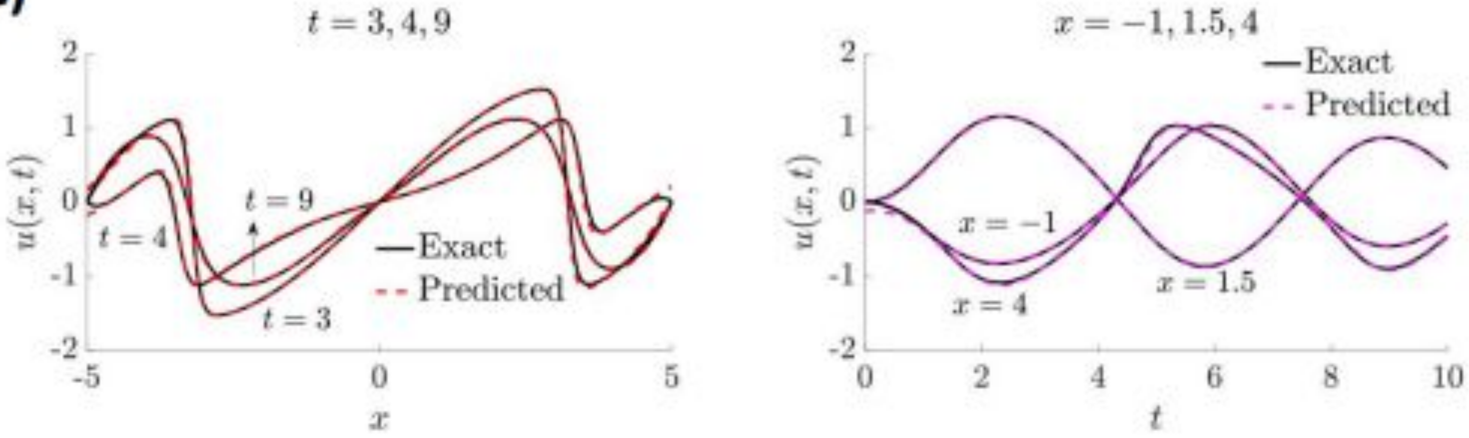
$$a = \sin(t),\ b = \sin(x),\ c = \cos(t) \text{ and } d = \cos(x)$$

If there is any source ?



If there is any source ?

(C)



Ground truth: $u_t + uu_x - 0.1u_{xx} = \sin(x)\sin(t)$

Discovered: $u_t + 1.002uu_x - 0.088u_{xx} = 0.995\sin(x)\sin(t)$

If we miss some terms ?

$$w_t = -uw_x - vw_y + 0.01w_{xx} + 0.01w_{yy}$$

If we miss some terms ?

$$w_t = -\cancel{v w_x} - v w_y + 0.01 w_{xx} + 0.01 w_{yy}$$

If we miss some terms ?

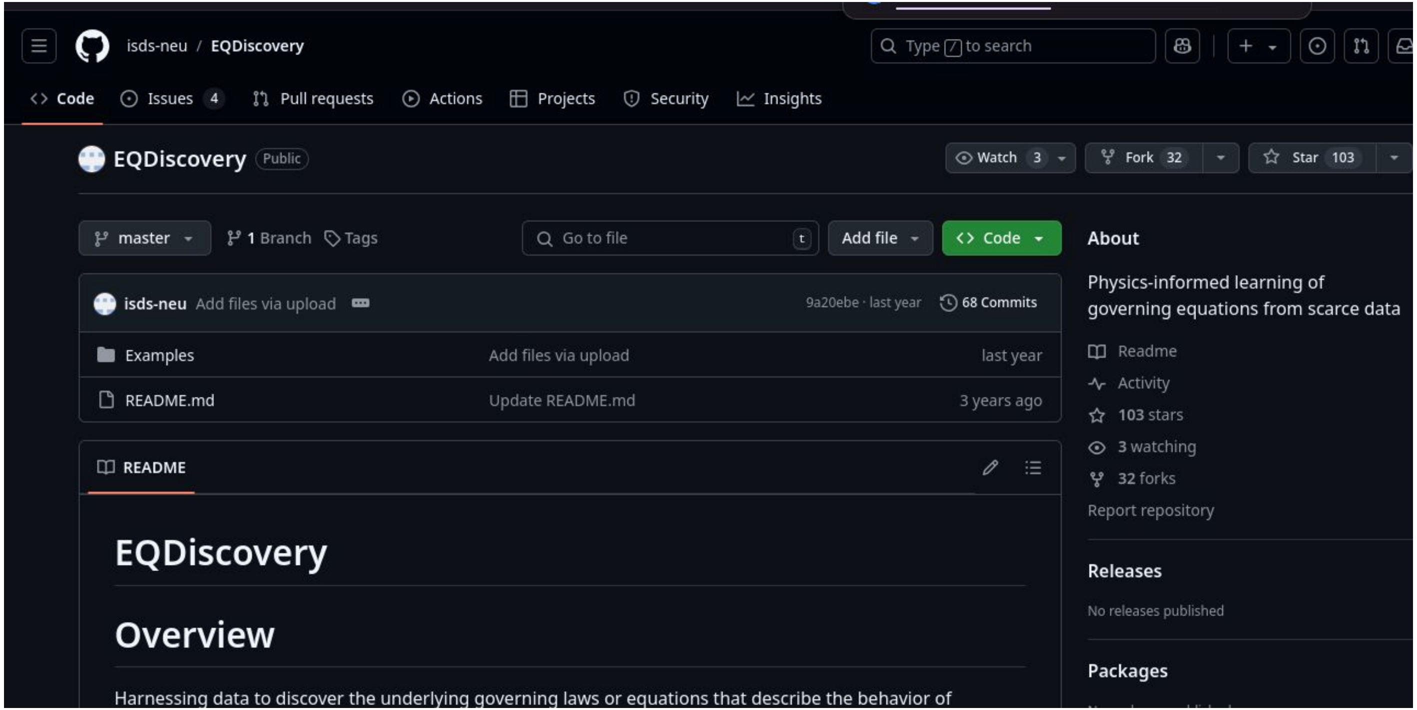
$$w_t = -\cancel{uw_x} - vw_y + 0.01w_{xx} + 0.01w_{yy}$$

$$w_t = -0.253w_x + 0.008w_{yy} + 0.035uw_{xx} - 0.782u^2w_x - 0.026u^2w_{xx} - 0.616vw_y - 0.155v^2w_x - 0.526uvw_y$$

Hyperparameters

Example		α[a]			γ	ADO				
		r_σ	Pre-training	ADO & Post-training		β[b]	ADO Iteration	Adam Epochs	STRidge Cycles	$\Delta\delta$[c]
Single Dataset	Burgers'	1.4	1	2	1E-7	$\mathcal{L}_p(\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Lambda}}_0; \mathcal{D}_c^{va})$	6	1000	100	1
	KS	19.4	1	10	1E-7	$\mathcal{L}_p(\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Lambda}}_0; \mathcal{D}_c^{va})$	6	1000	100	1
	Schrödinger	0.05	0.1	0.5	1E-7	$100\mathcal{L}_p(\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Lambda}}_0; \mathcal{D}_c^{va})$	6	1000	100	100
	NS	1.7	1	2	1E-7	$\mathcal{L}_p(\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Lambda}}_0; \mathcal{D}_c^{va})$	6	1000	100	1
	$\lambda - \omega$ RD	1.4	10	10	1E-7	$\mathcal{L}_p(\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Lambda}}_0; \mathcal{D}_c^{va})$	6	1000	100	1
Multiple Datasets	Burgers'	0.02	0.01	0.1	1E-7	$\mathcal{L}_p(\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Lambda}}_0; \mathcal{D}_c^{va})$	6	1000	100	1
	FN RD	17.2	1	10	1E-7	$\mathcal{L}_p(\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Lambda}}_0; \mathcal{D}_c^{va})$	6	1000	100	1
Experimental	Cell	2.2E3	200	2.2E3	1E-7	$\mathcal{L}_p(\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Lambda}}_0; \mathcal{D}_c^{va})$	6	1000	100	1

Github



https://github.com/isds-neu/EQDiscovery

Github

isds-neu Add files via upload 9a20ebe · last year History

Name	Last commit message	Last commit date
..		
Discovery with Experimental Datasets	Add files via upload	3 years ago
Discovery with Multiple Datasets	Add files via upload	3 years ago
Discovery with Single Dataset	Add files via upload	last year
Discussion	Add files via upload	3 years ago

https://github.com/isds-neu/EQDiscovery

Github

Name	Last commit message	Last commit date
..		
__pycache__	Add files via upload	3 years ago
Burgers_CubeIC_new.mat	Add files via upload	3 years ago
Burgers_GaussIC_new.mat	Add files via upload	3 years ago
Burgers_SineIC_new.mat	Add files via upload	3 years ago
Burgers_UniformSetting_Pre_ADO.py	Add files via upload	3 years ago
Burgers_UniformSetting_Pt.py	Add files via upload	3 years ago

https://github.com/isds-neu/EQDiscovery

Images from bing